Comprehensive Kernel Instrumentation via Dynamic Binary Translation

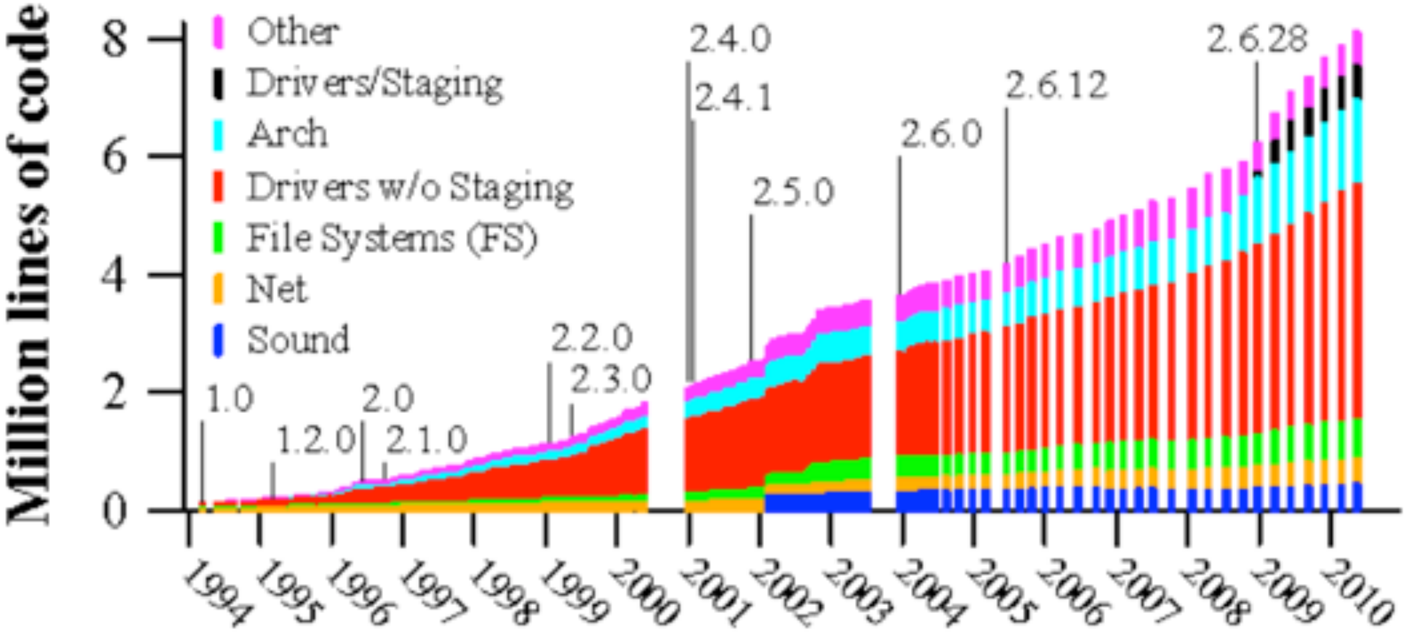
Peter Feiner

Angela Demke Brown

Ashvin Goel

University of Toronto

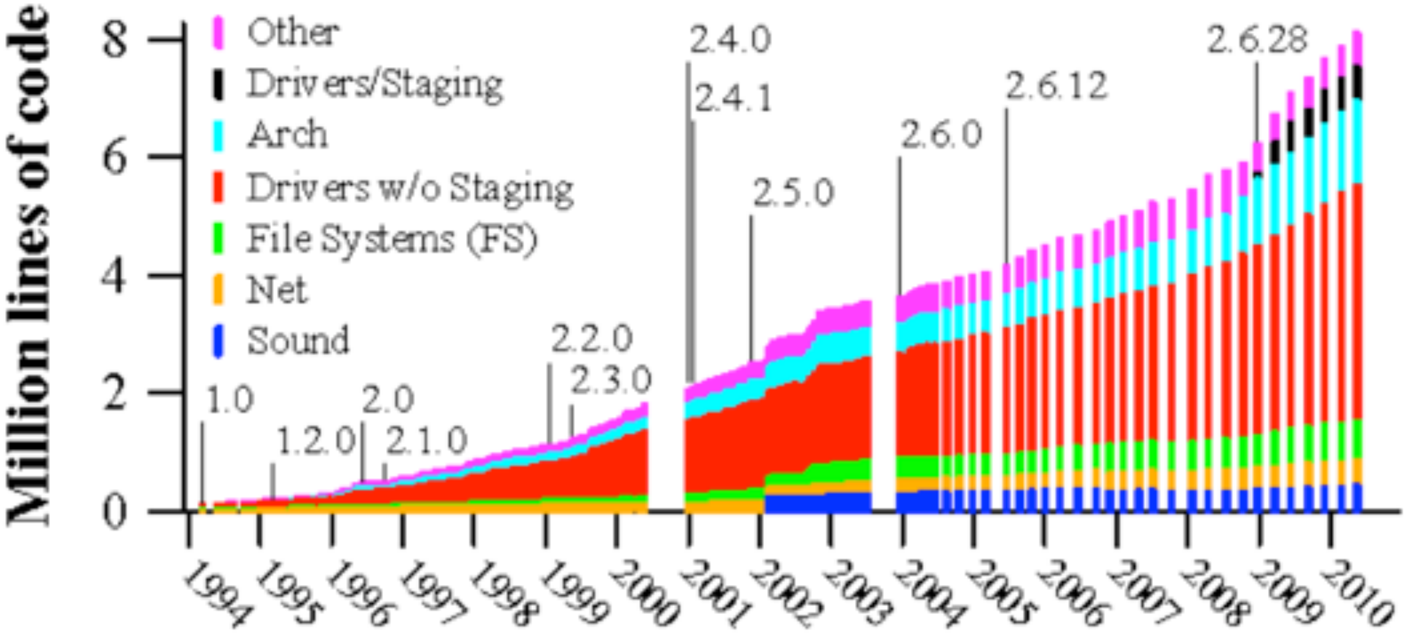
Complexity of Operating Systems



Complexity of Operating Systems

Growth in code size

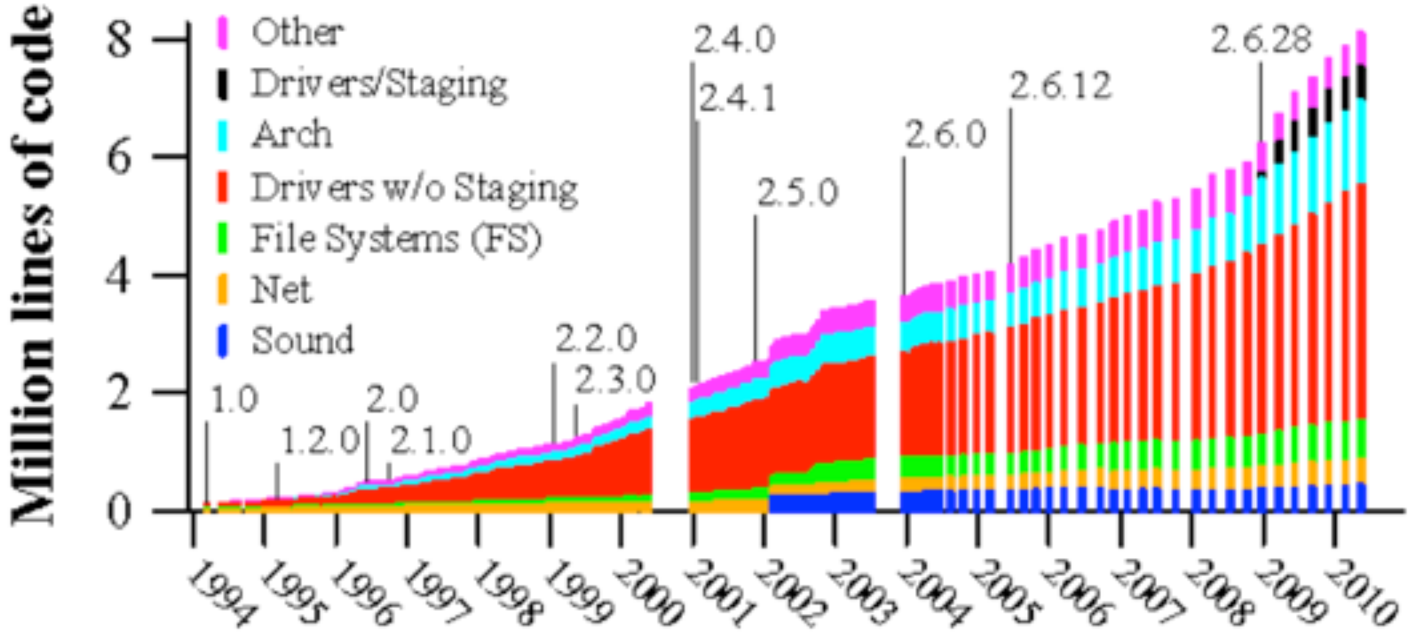
- Palix, ASPLOS 2011
- Many new drivers!



Complexity of Operating Systems

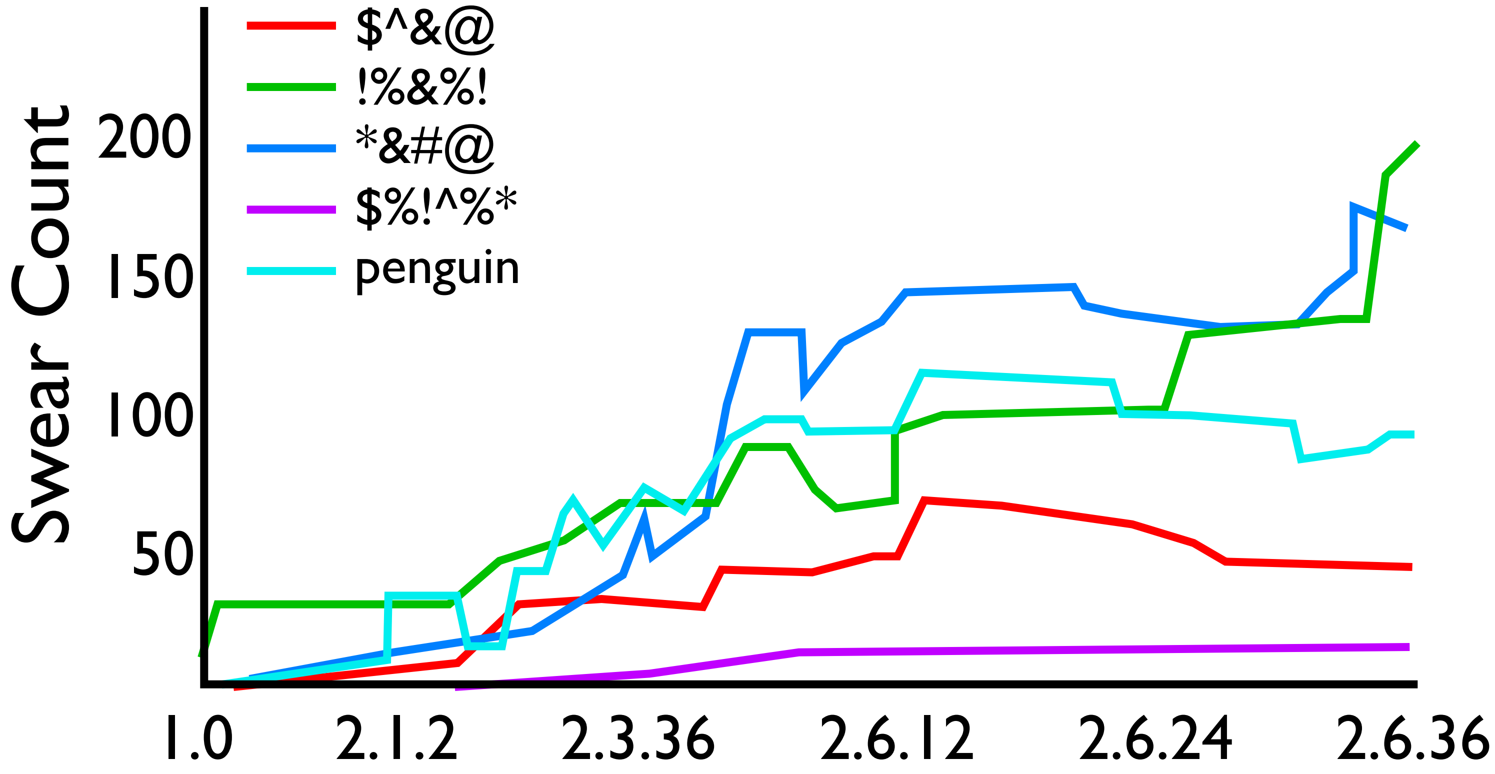
Growth in code size

- Palix, ASPLOS 2011
- Many new drivers!



More swearing

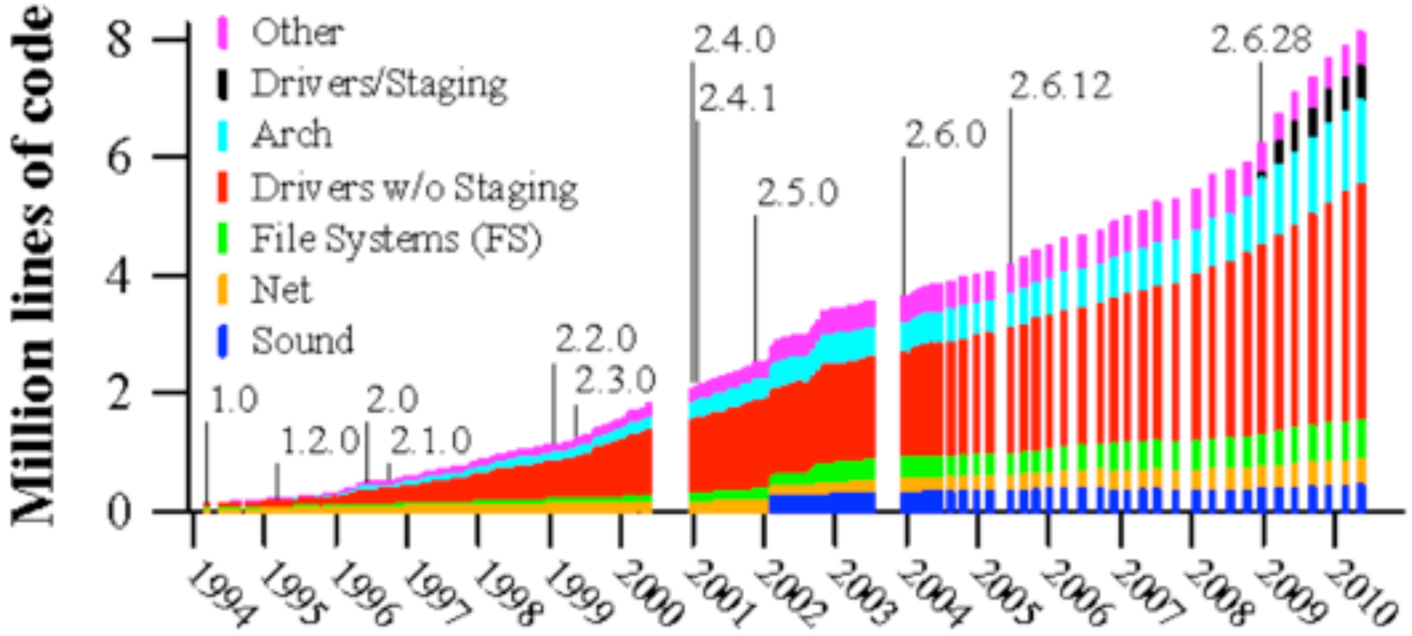
- Vidar Holen, 2012



Complexity of Operating Systems

Growth in code size

- Palix, ASPLOS 2011
- Many new drivers!

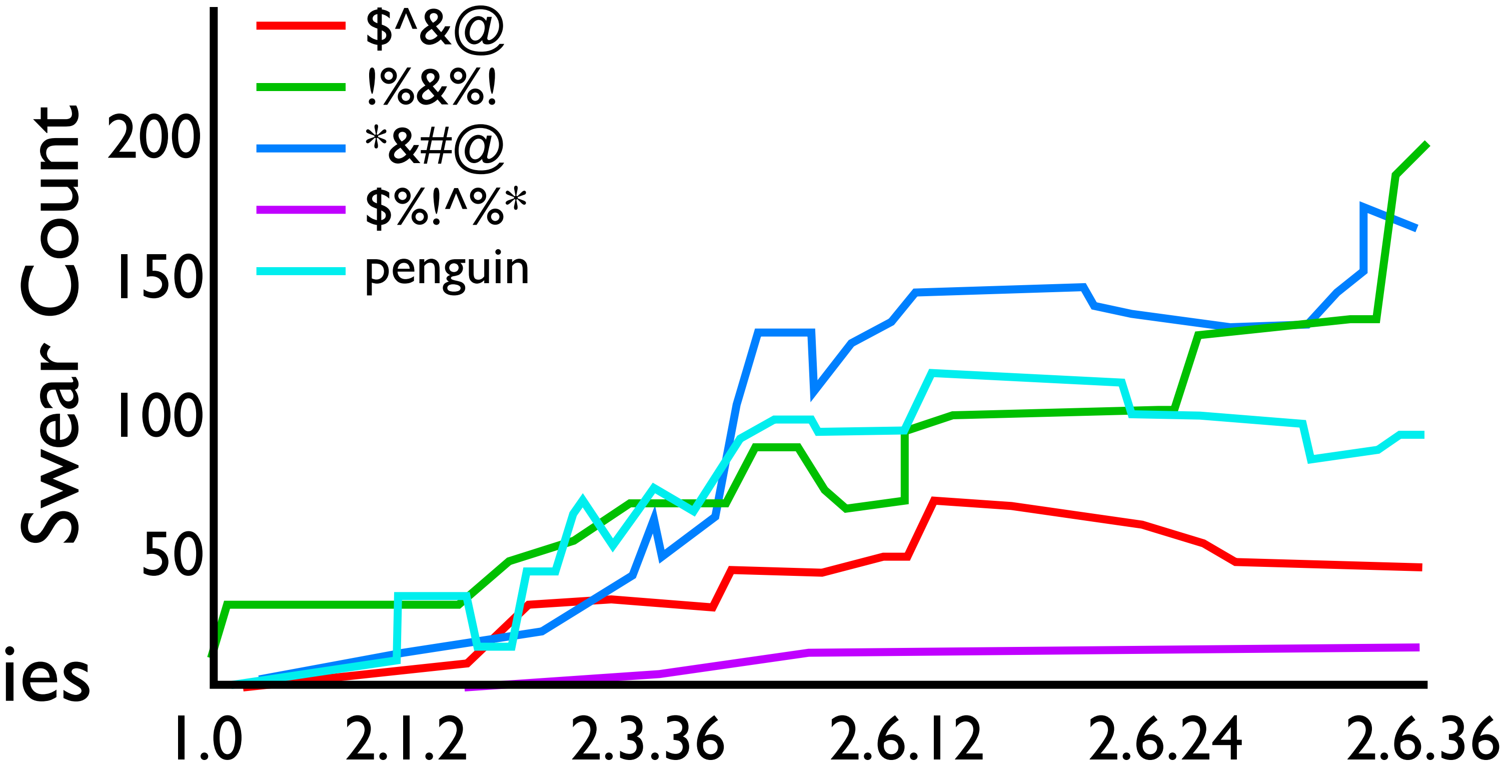


More swearing

- Vidar Holen, 2012

Bugs are inevitable!

- Need coping strategies



Tools would be nice

Awesome tools for **user** code

- Memcheck
- Program Shepherding

Use Dynamic Binary Translation (DBT)

- Rewrite binaries as they execute
- No need for source

Frameworks make building DBT tools easy

- DynamoRIO, Valgrind, Pin

No framework for **OS** code

Our Framework

Our Framework

Ported DynamRIO to Linux kernel

- Runs on bare metal

Our Framework

Ported DynamRIO to Linux kernel

- Runs on bare metal

Port took **18 Months**

Our Framework

Ported DynamRIO to Linux kernel

- Runs on bare metal

Port took **18 Months**

Built OS debugging tools in **5 days**

- Heap debugging
  - Use after free
  - Heap corruption
- Stack overflow monitor

Our Framework

Ported DynamRIO to Linux kernel

- Runs on bare metal

Port took **18 Months**

Built OS debugging tools in **5 days**

- Heap debugging
  - Use after free
  - Heap corruption
- Stack overflow monitor

Practical

- One author ran system on her desktop for 1 month

What About Hypervisors?

What About Hypervisors?

VMWare can use DBT on guests

- No instrumentation API

What About Hypervisors?

VMWare can use DBT on guests

- No instrumentation API

PinOS has instrumentation API

- PinOS = Pin + Xen
- Guest needs emulated devices
  - Useless for most driver code

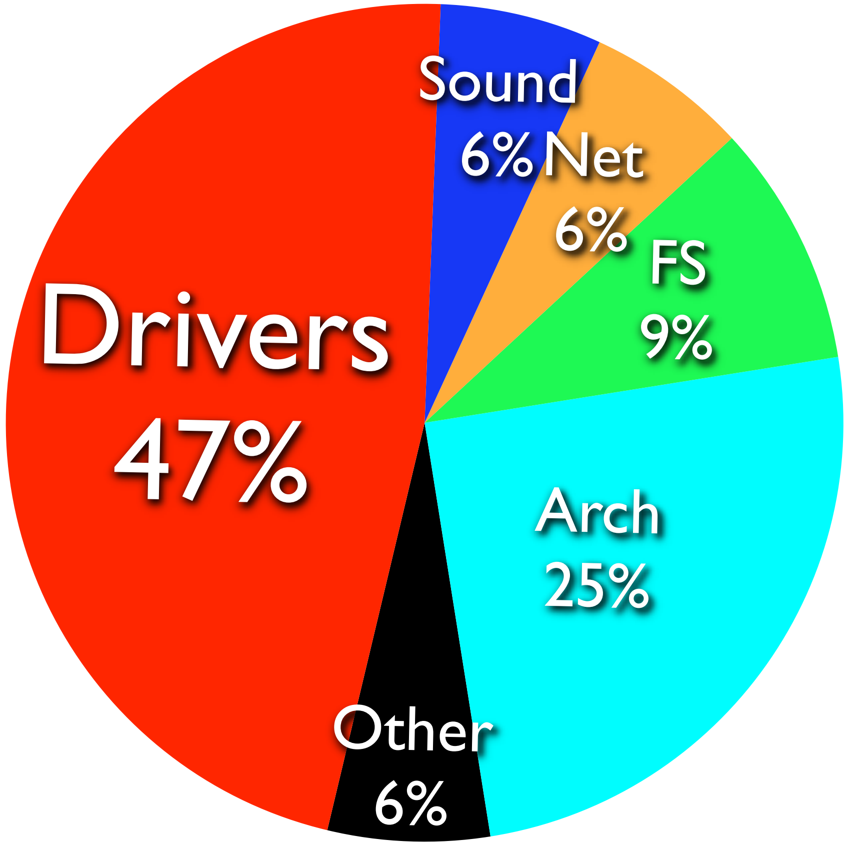
What About Hypervisors?

VMWare can use DBT on guests

- No instrumentation API

PinOS has instrumentation API

- PinOS = Pin + Xen
- Guest needs emulated devices
  - Useless for most driver code



Palix, ASPLOS 2011

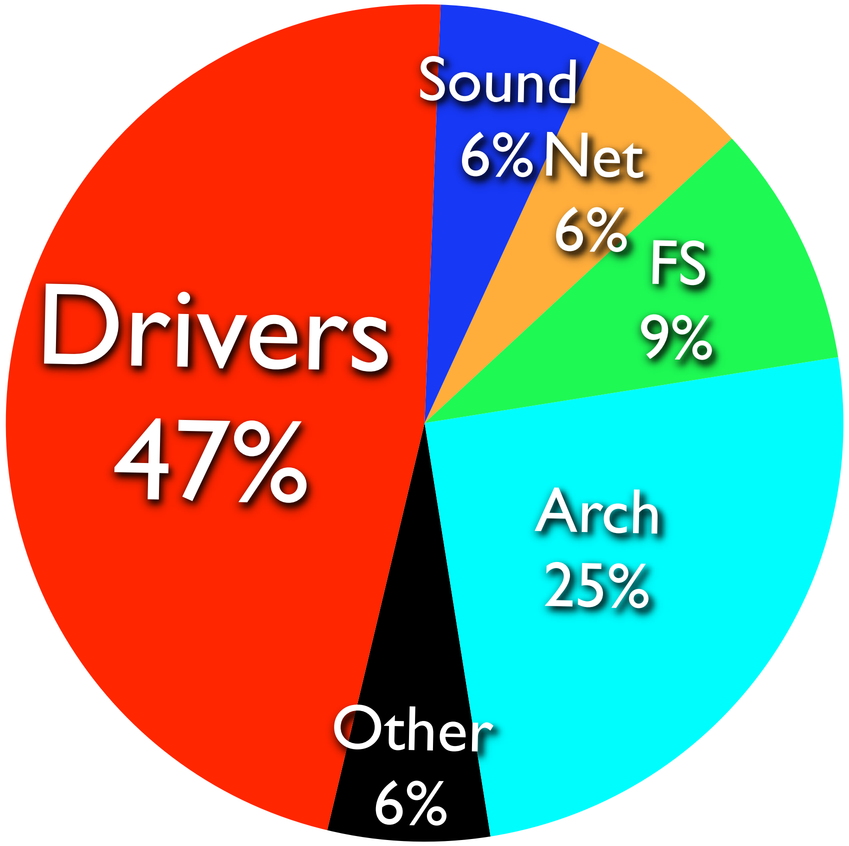
What About Hypervisors?

VMWare can use DBT on guests

- No instrumentation API

PinOS has instrumentation API

- PinOS = Pin + Xen
- Guest needs emulated devices
  - Useless for most driver code



Palix, ASPLOS 2011

So add DBT to a hypervisor with pass-through devices?

- Then you'd have the problems we show you how to solve
- ... problems with interrupts!

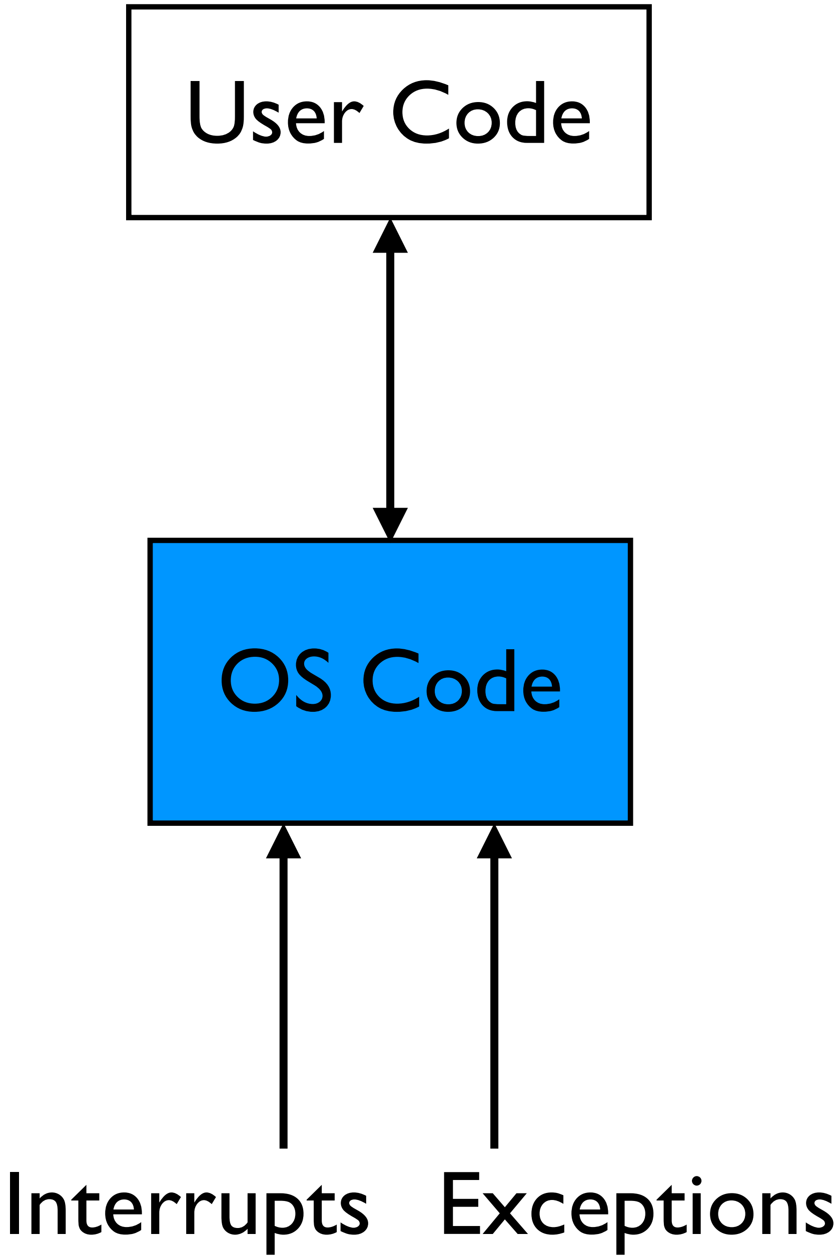
Framework Overview

1. Boot normally

2. Take over

3. JIT the OS

x86 → instrumented x86



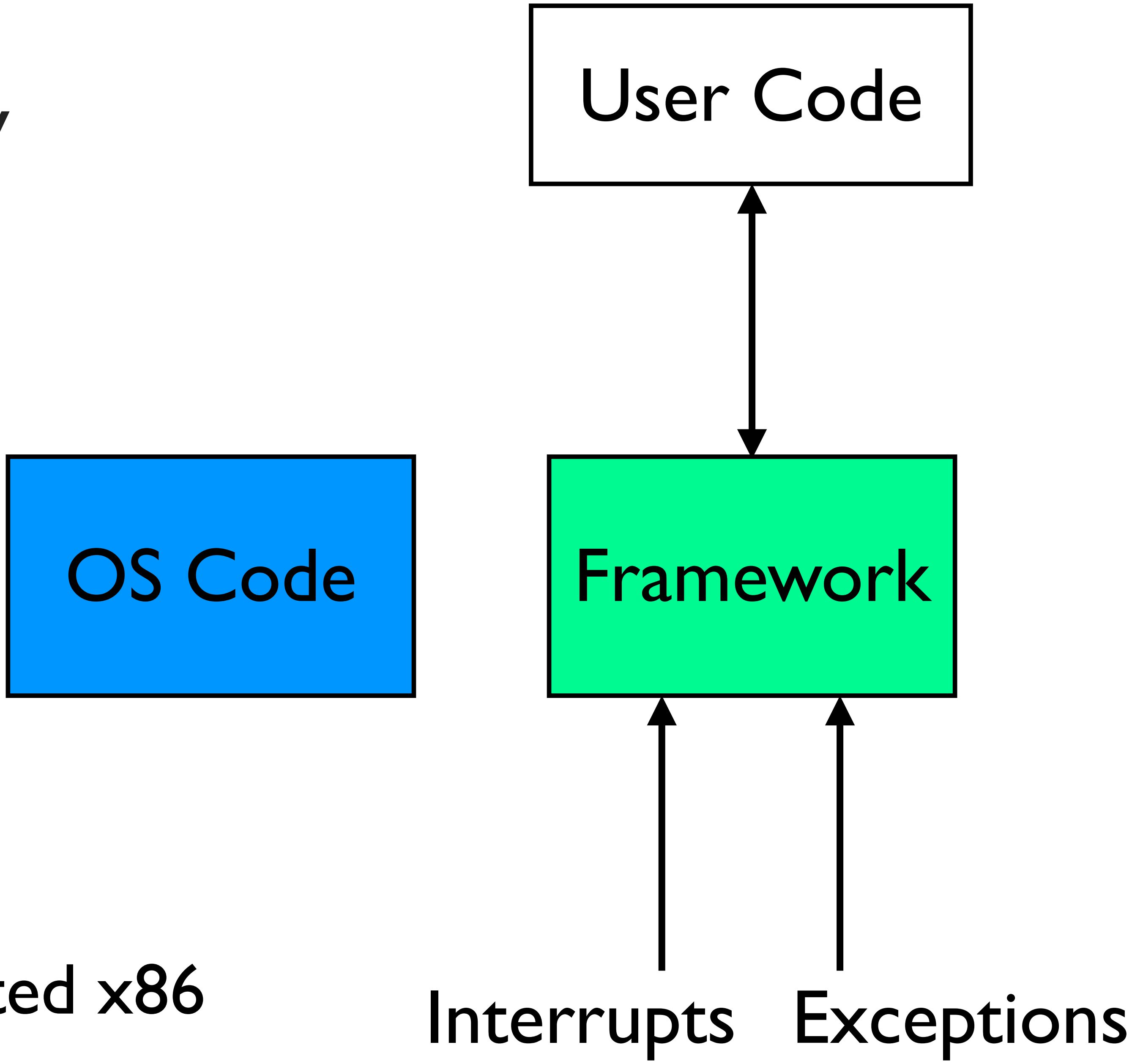
Framework Overview

1. Boot normally

2. Take over

3. JIT the OS

x86 → instrumented x86



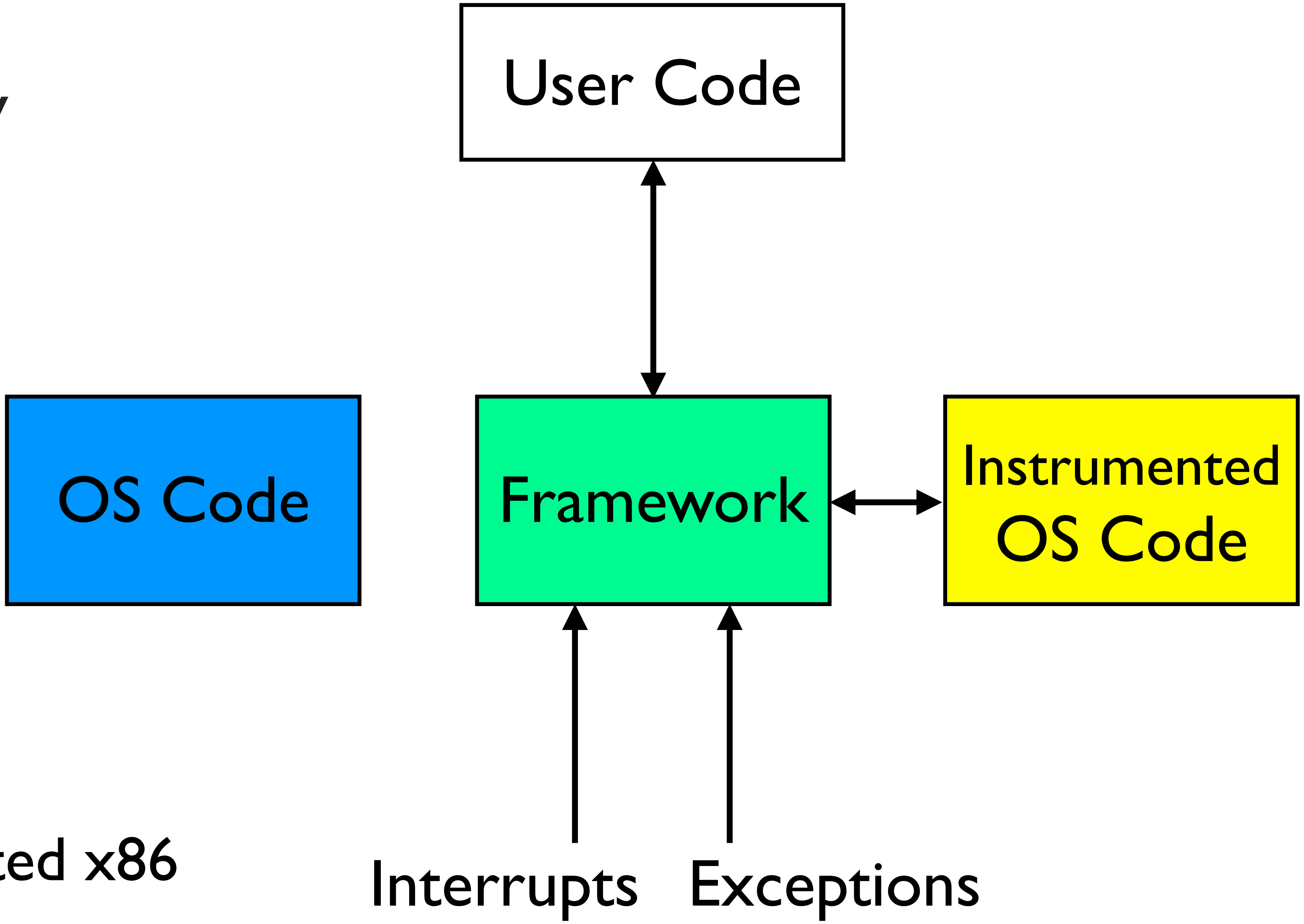
Framework Overview

1. Boot normally

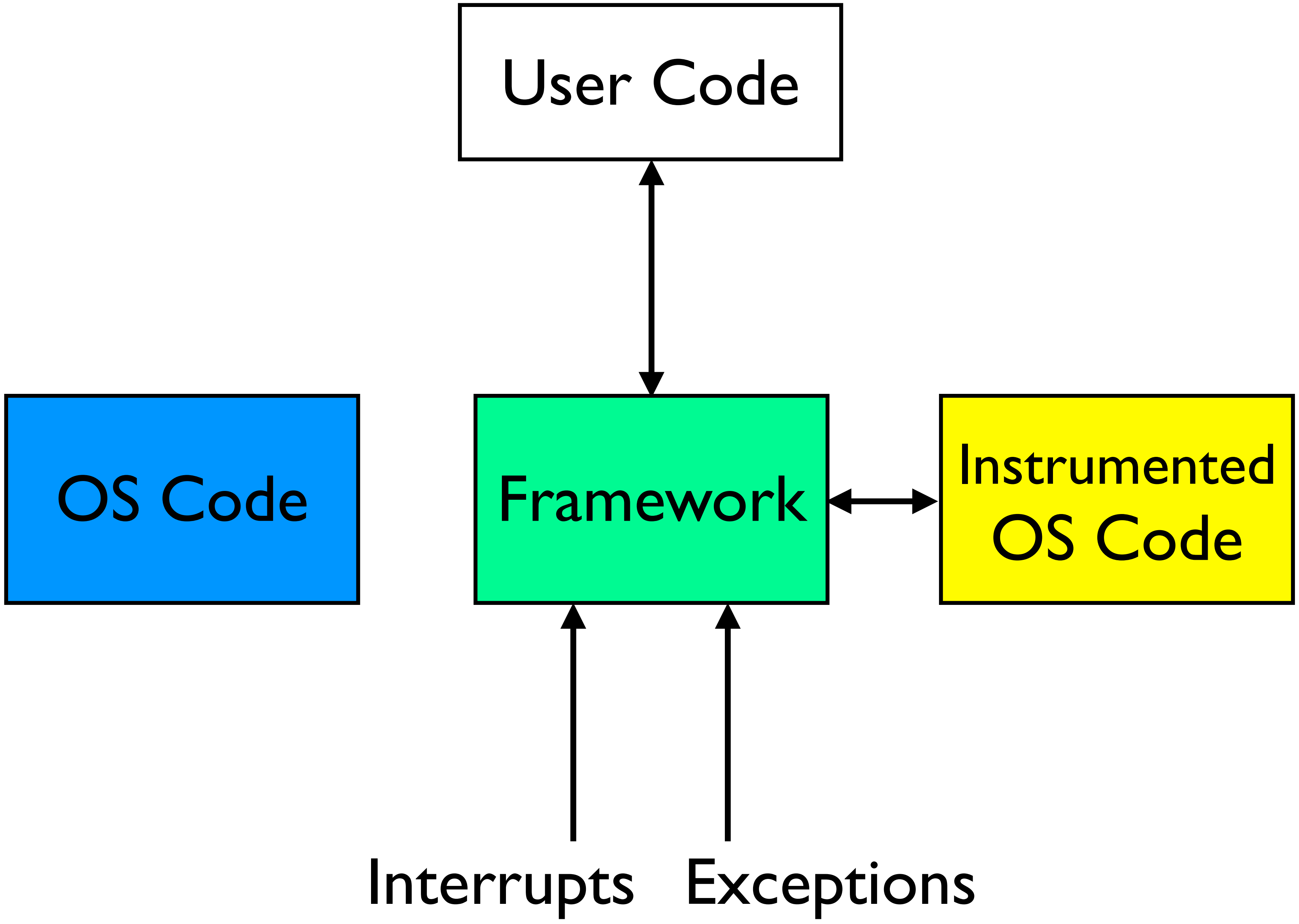
2. Take over

3. JIT the OS

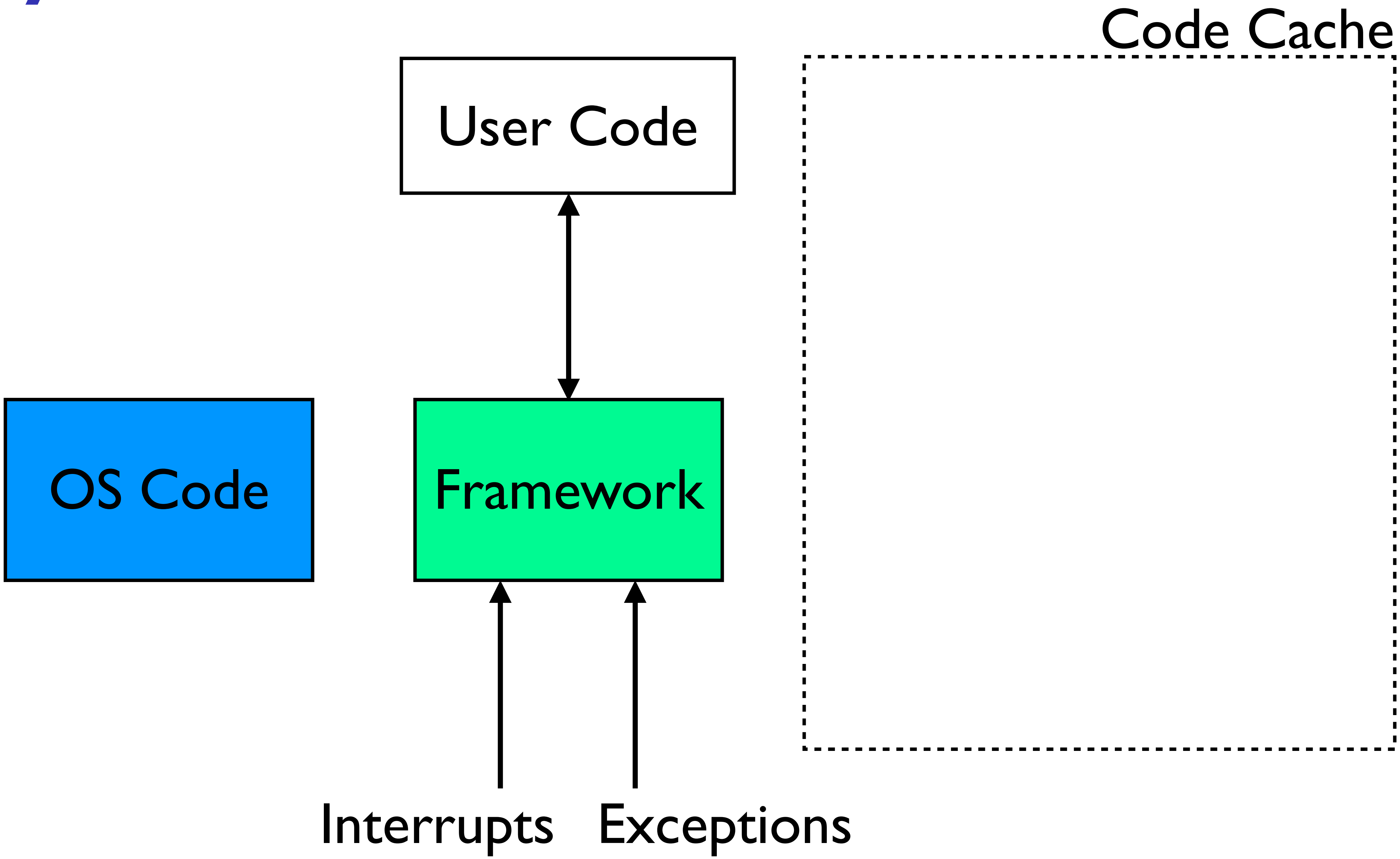
x86 → instrumented x86



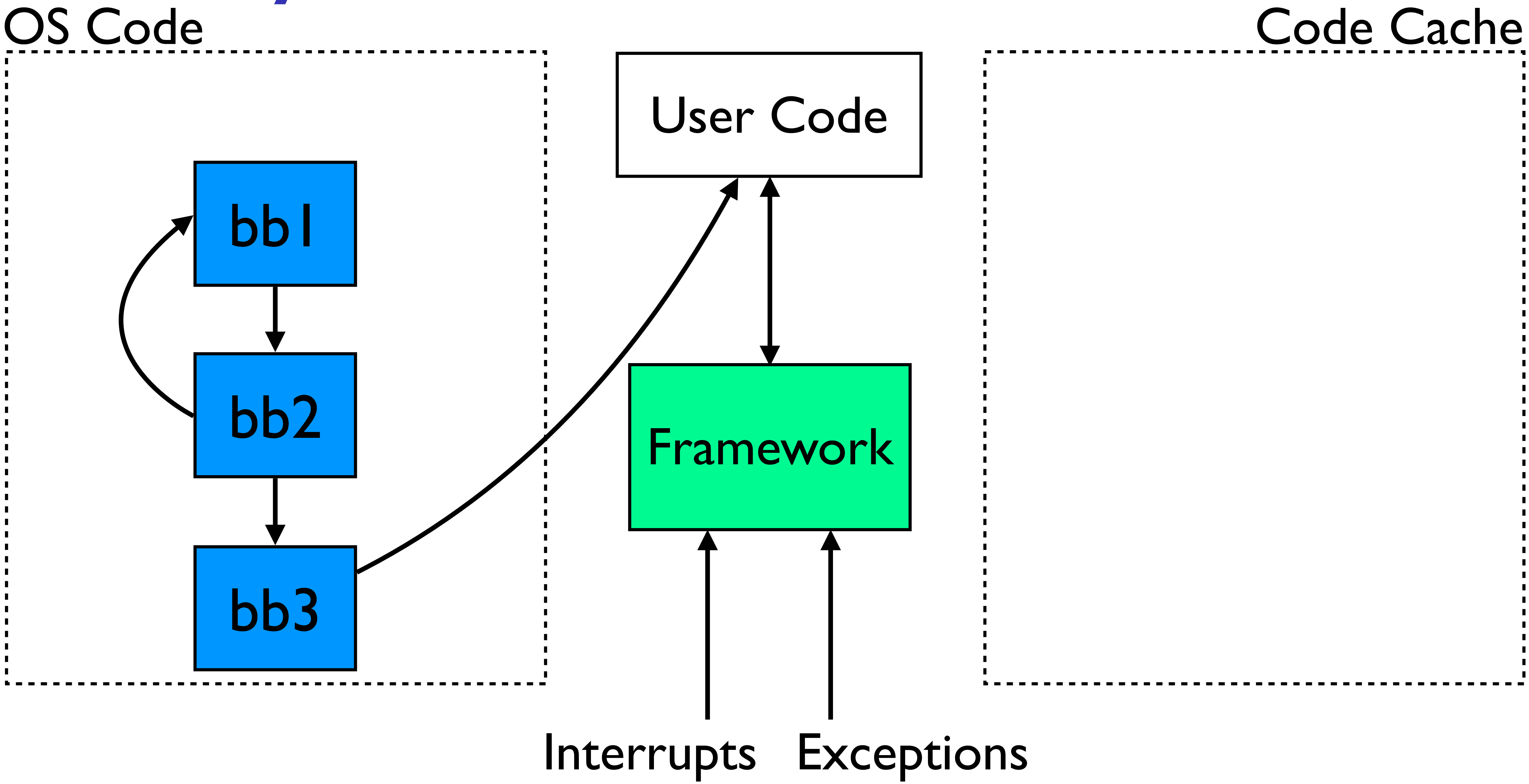
Dynamic Instrumentation



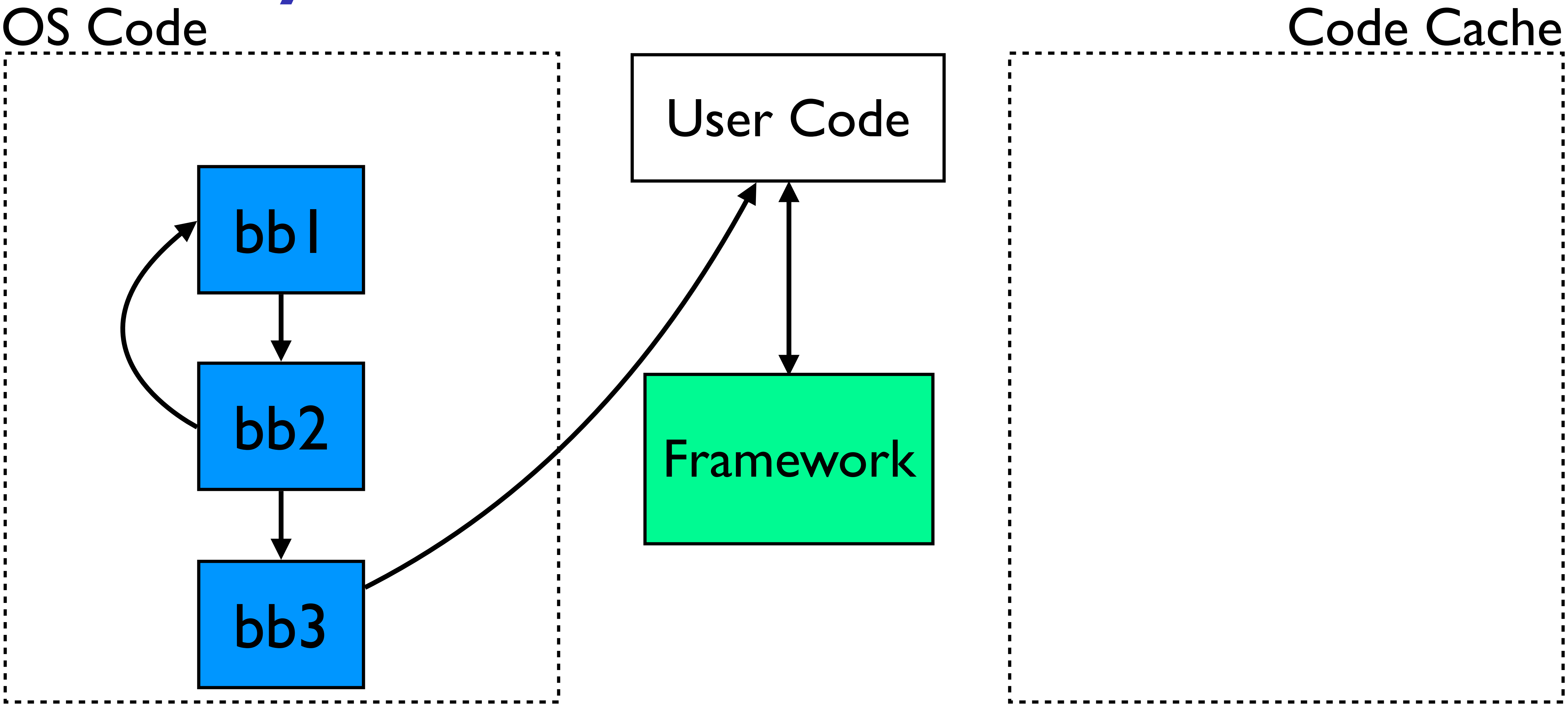
Dynamic Instrumentation



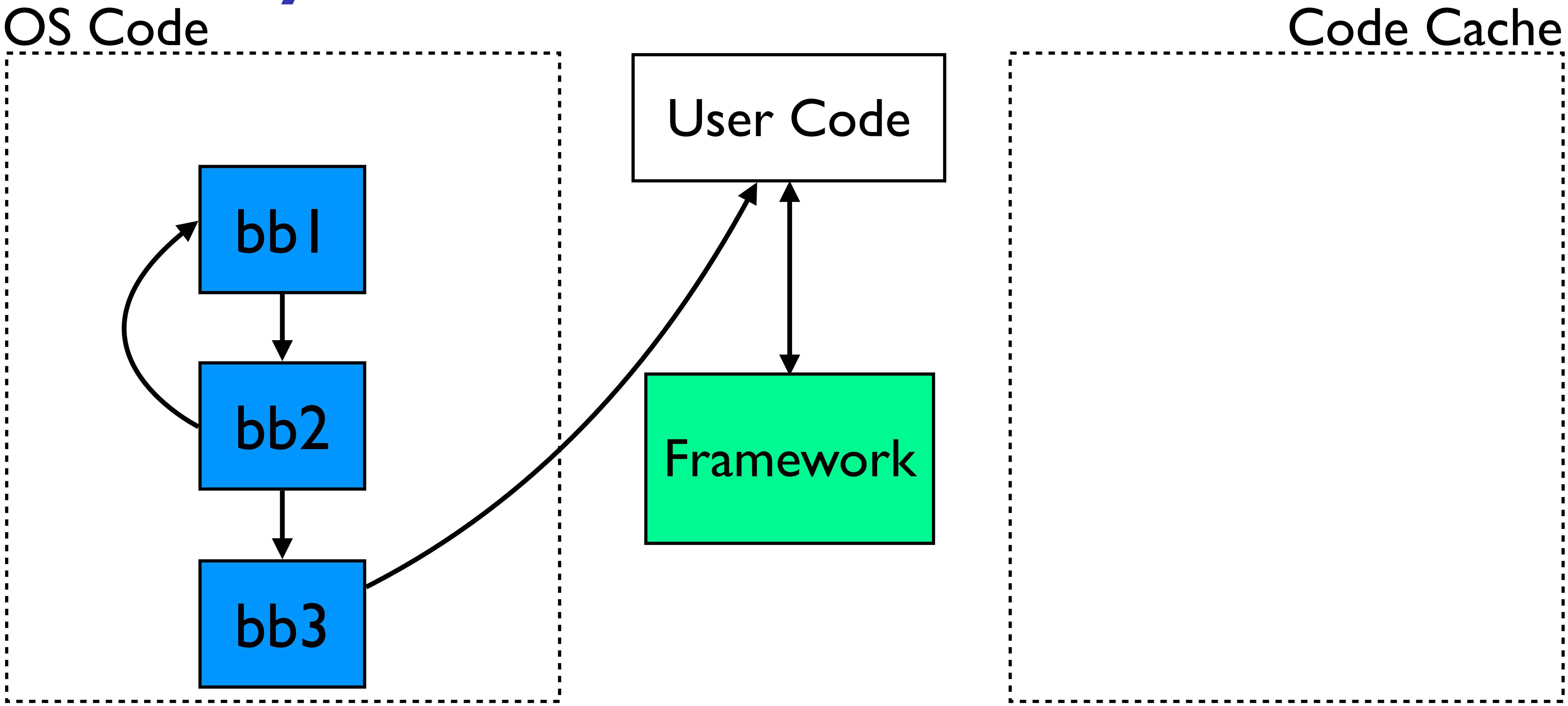
Dynamic Instrumentation



Dynamic Instrumentation



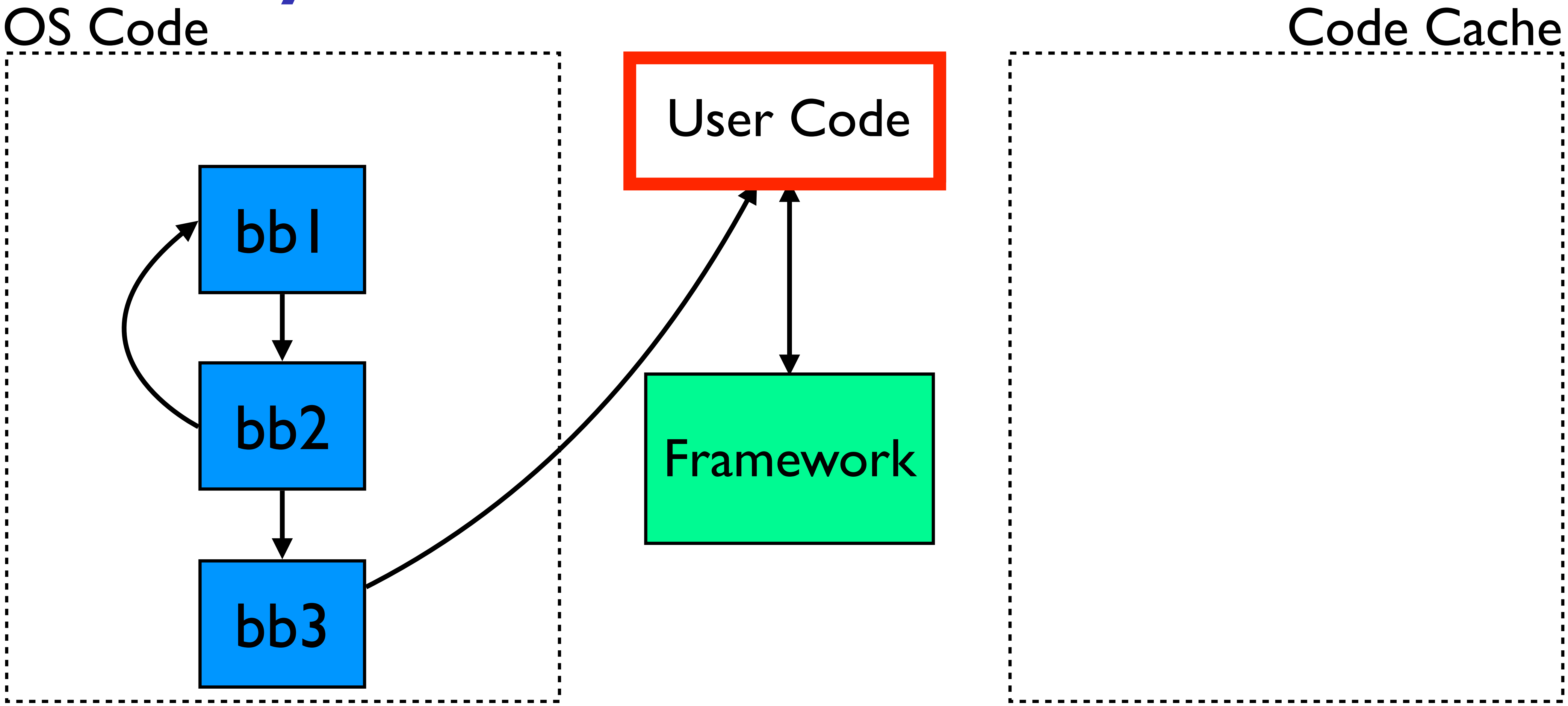
Dynamic Instrumentation



System call example

- bb1 is entry point

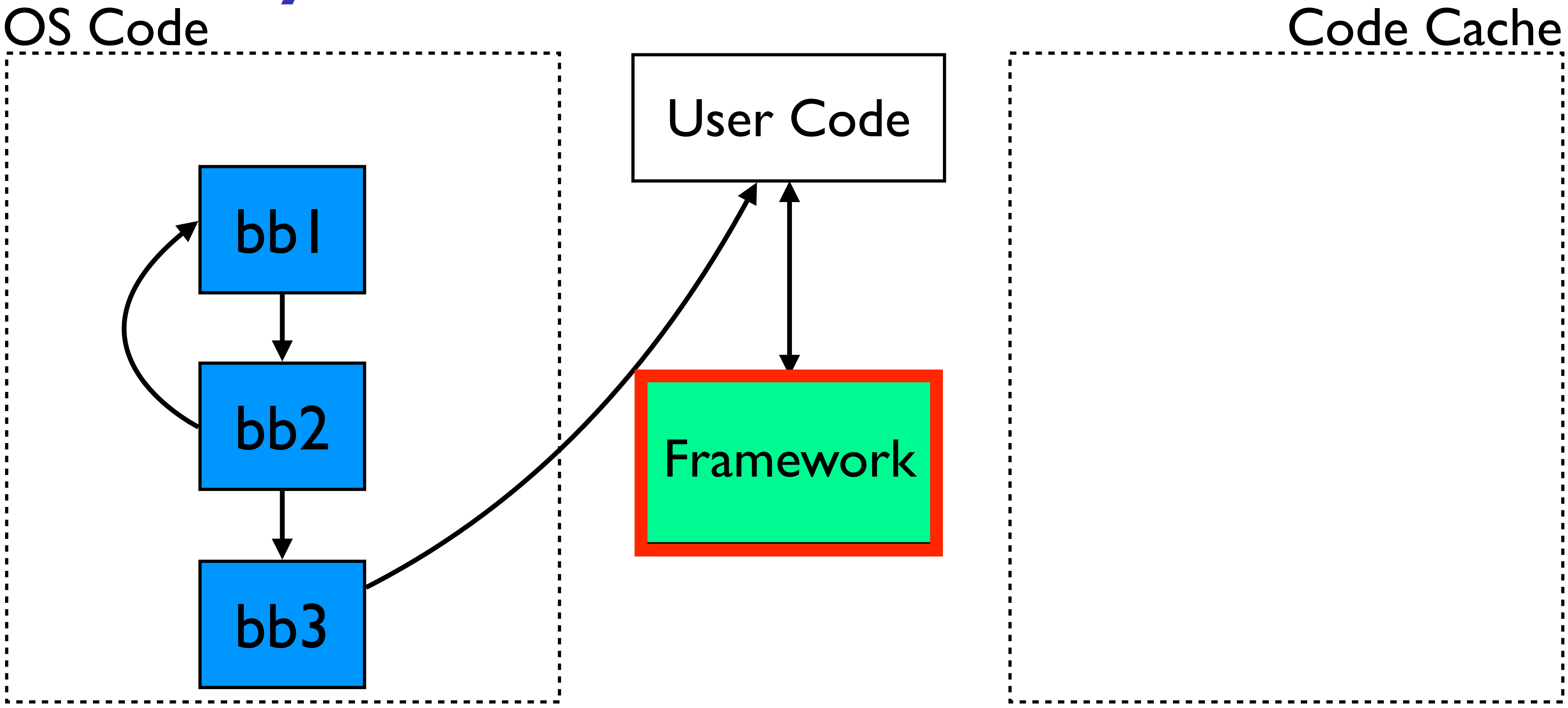
Dynamic Instrumentation



System call example

- bb1 is entry point

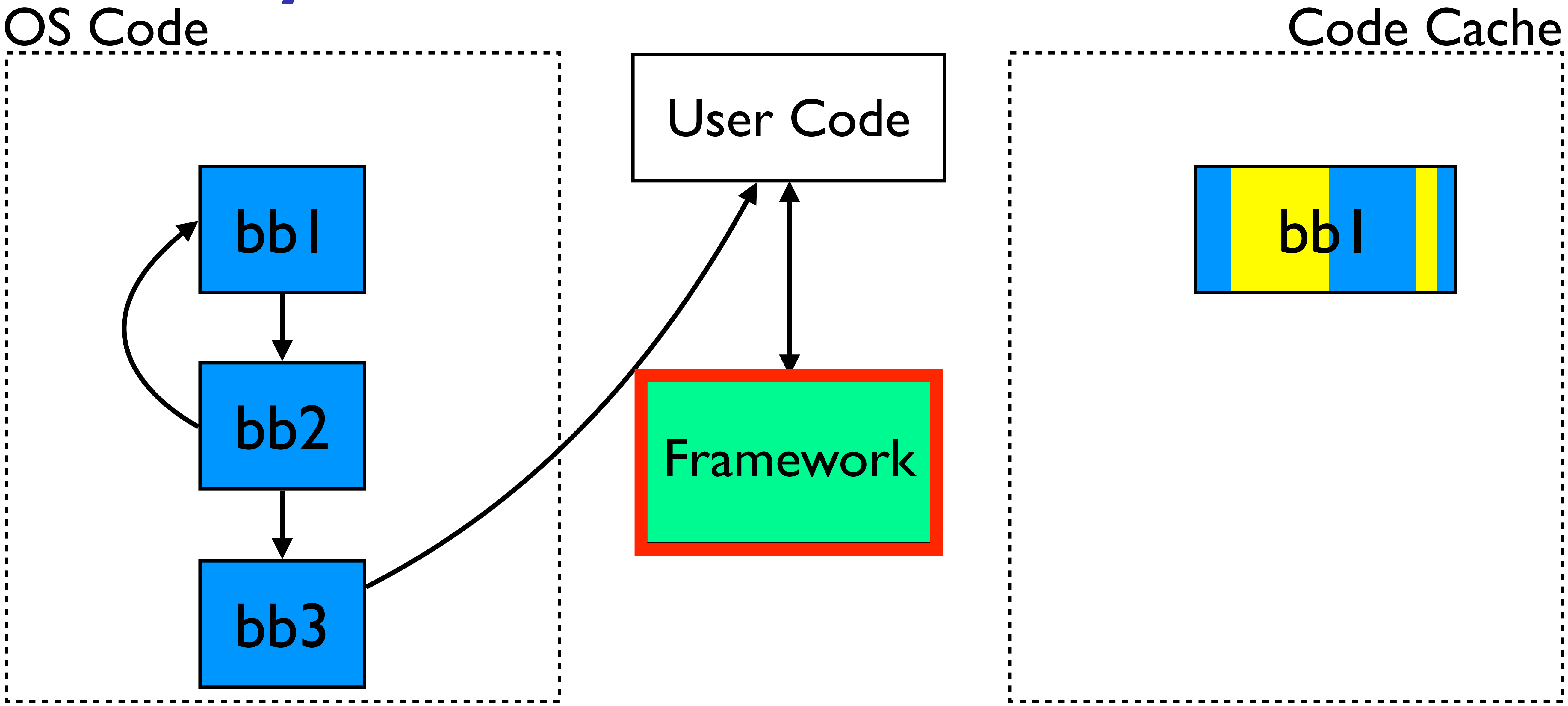
Dynamic Instrumentation



System call example

- bb1 is entry point

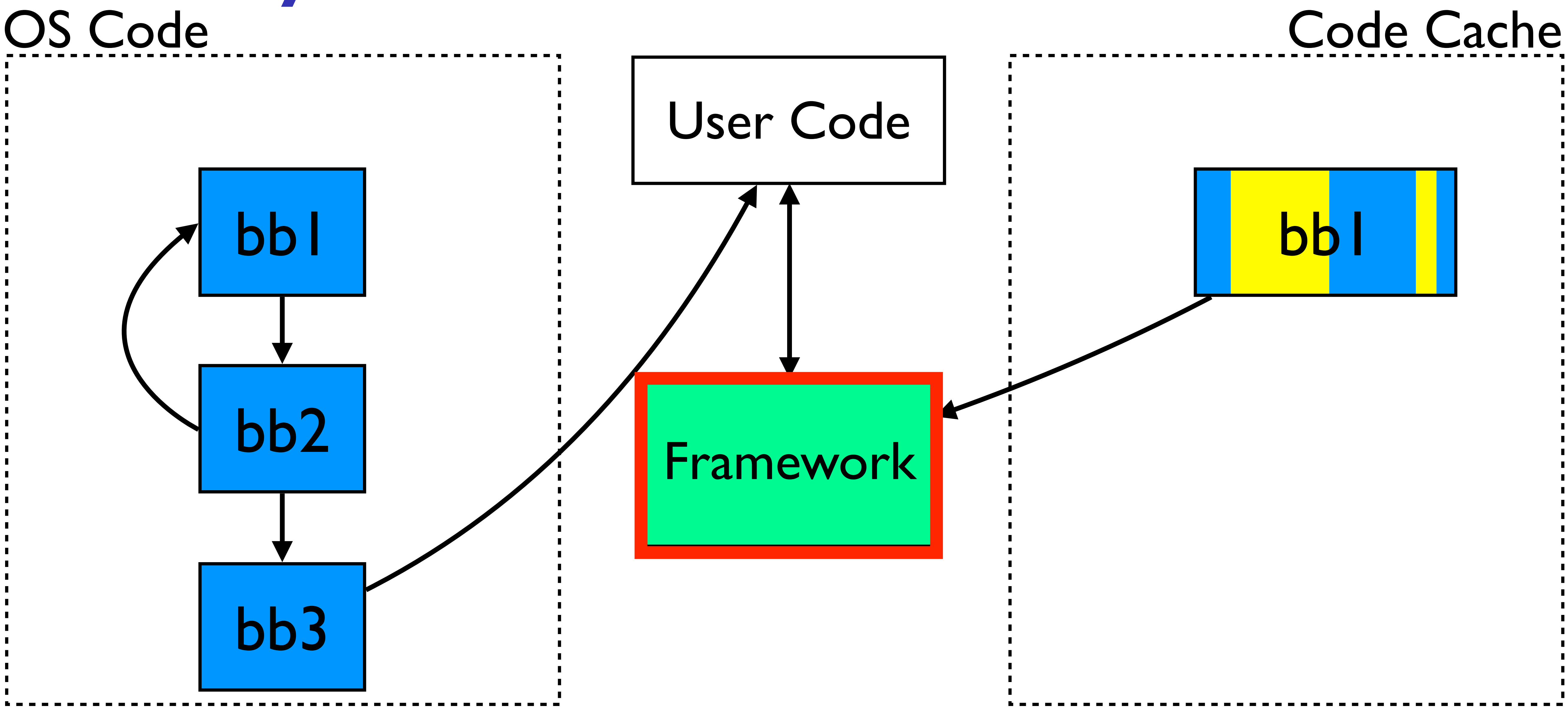
Dynamic Instrumentation



System call example

- bb1 is entry point

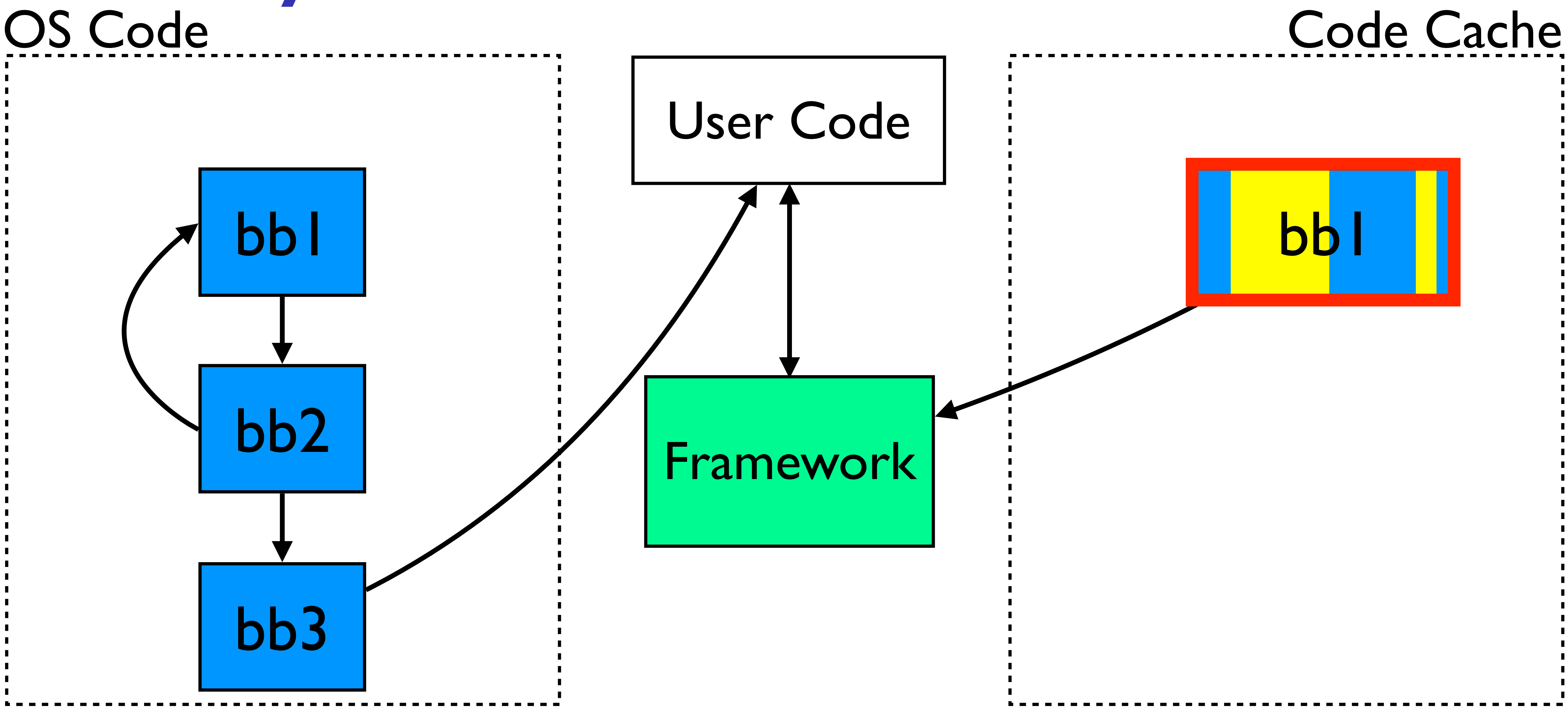
Dynamic Instrumentation



System call example

- bb1 is entry point

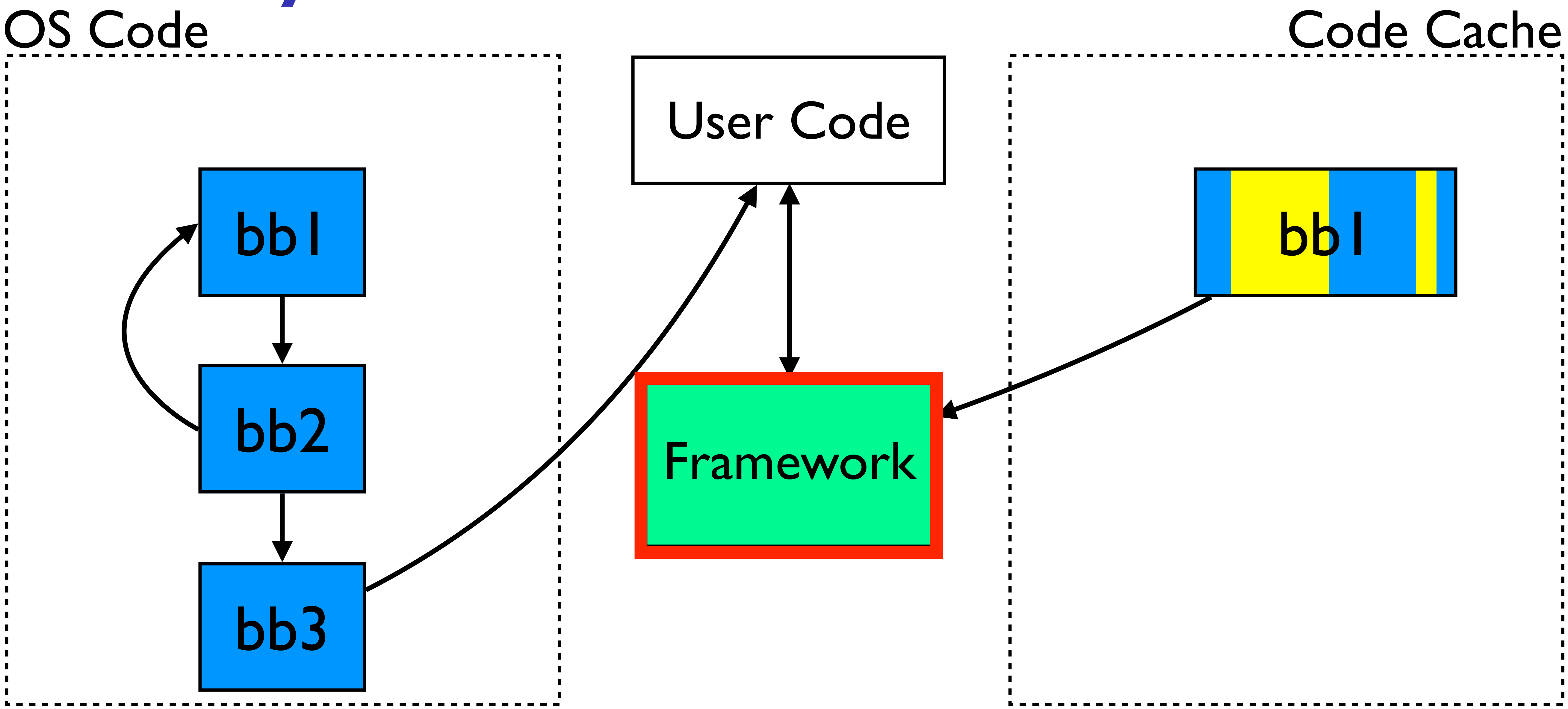
Dynamic Instrumentation



System call example

- bb1 is entry point

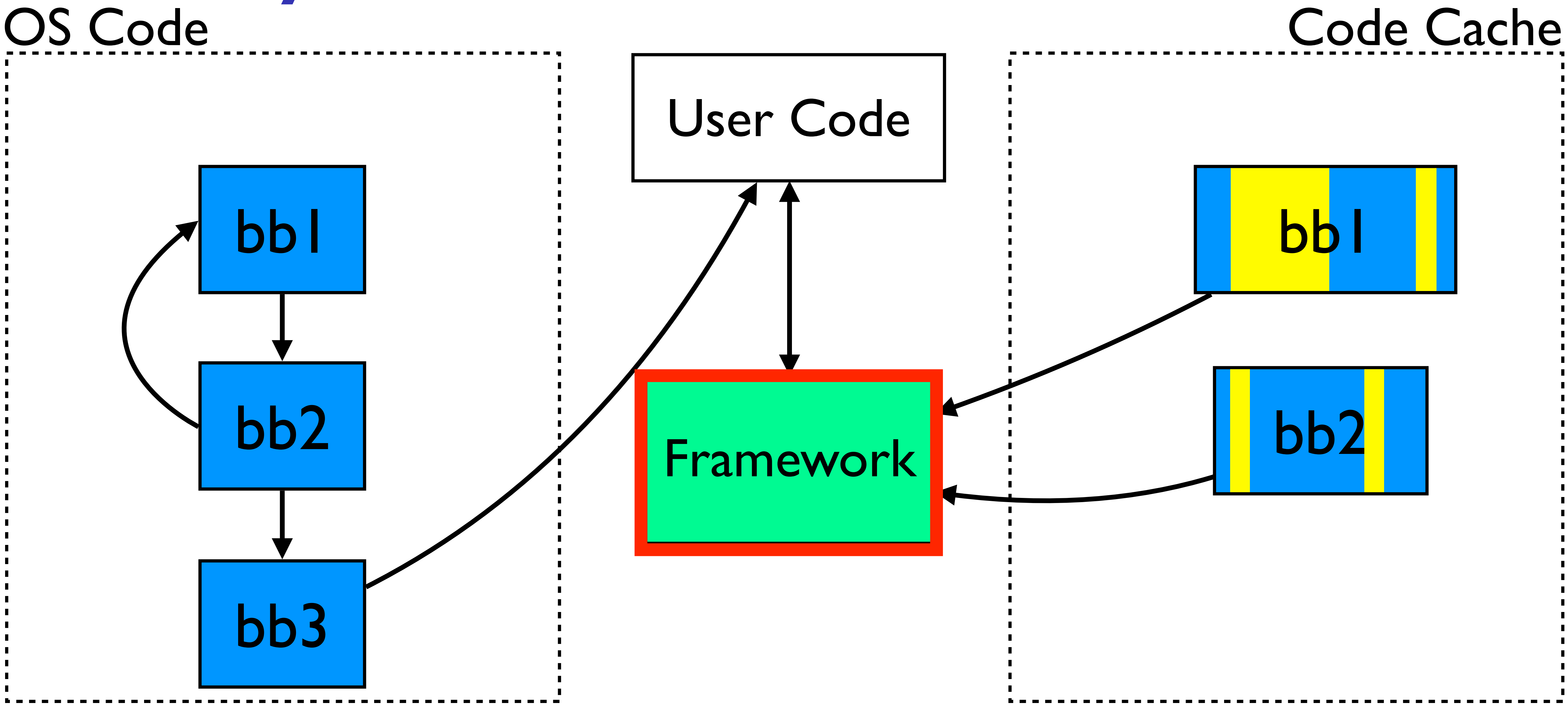
Dynamic Instrumentation



System call example

- bb1 is entry point

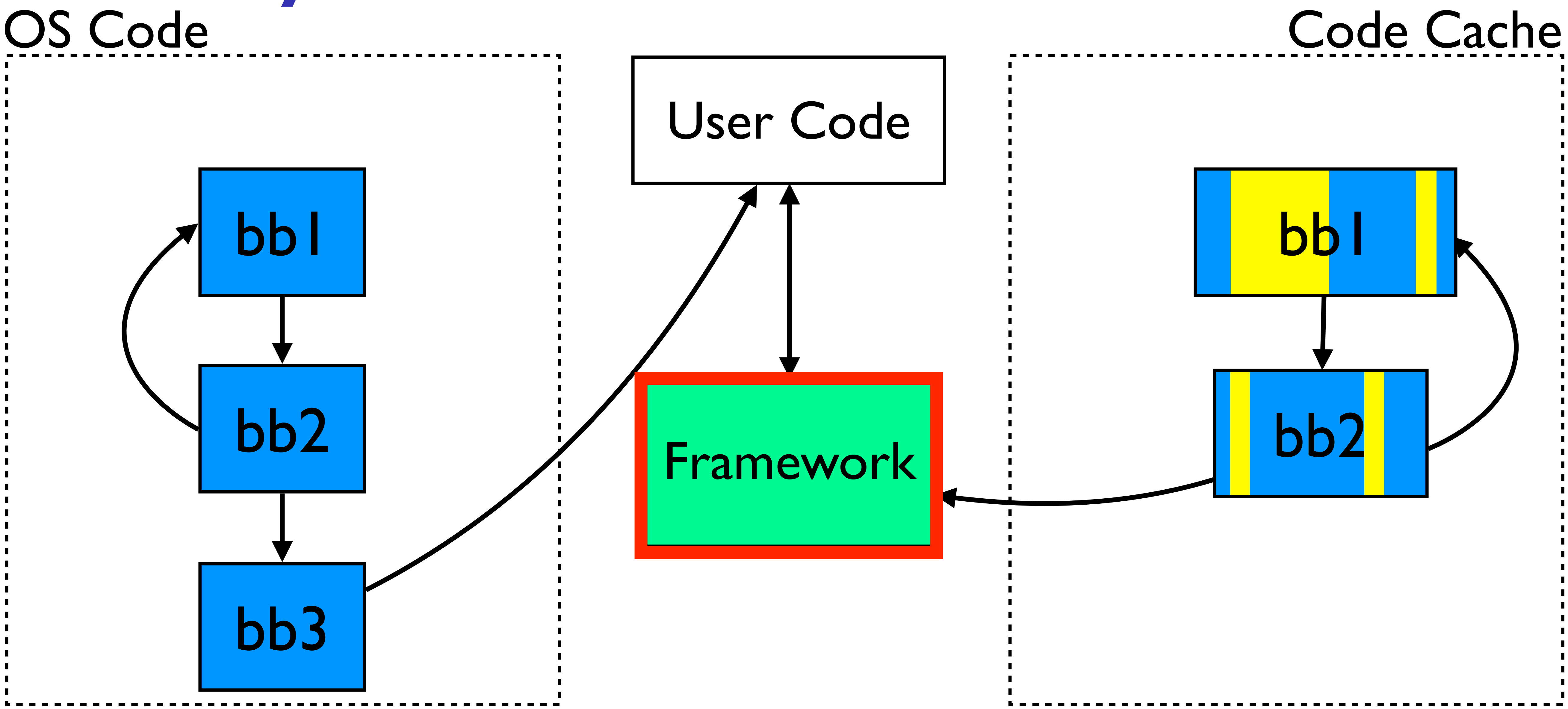
Dynamic Instrumentation



System call example

- bb1 is entry point

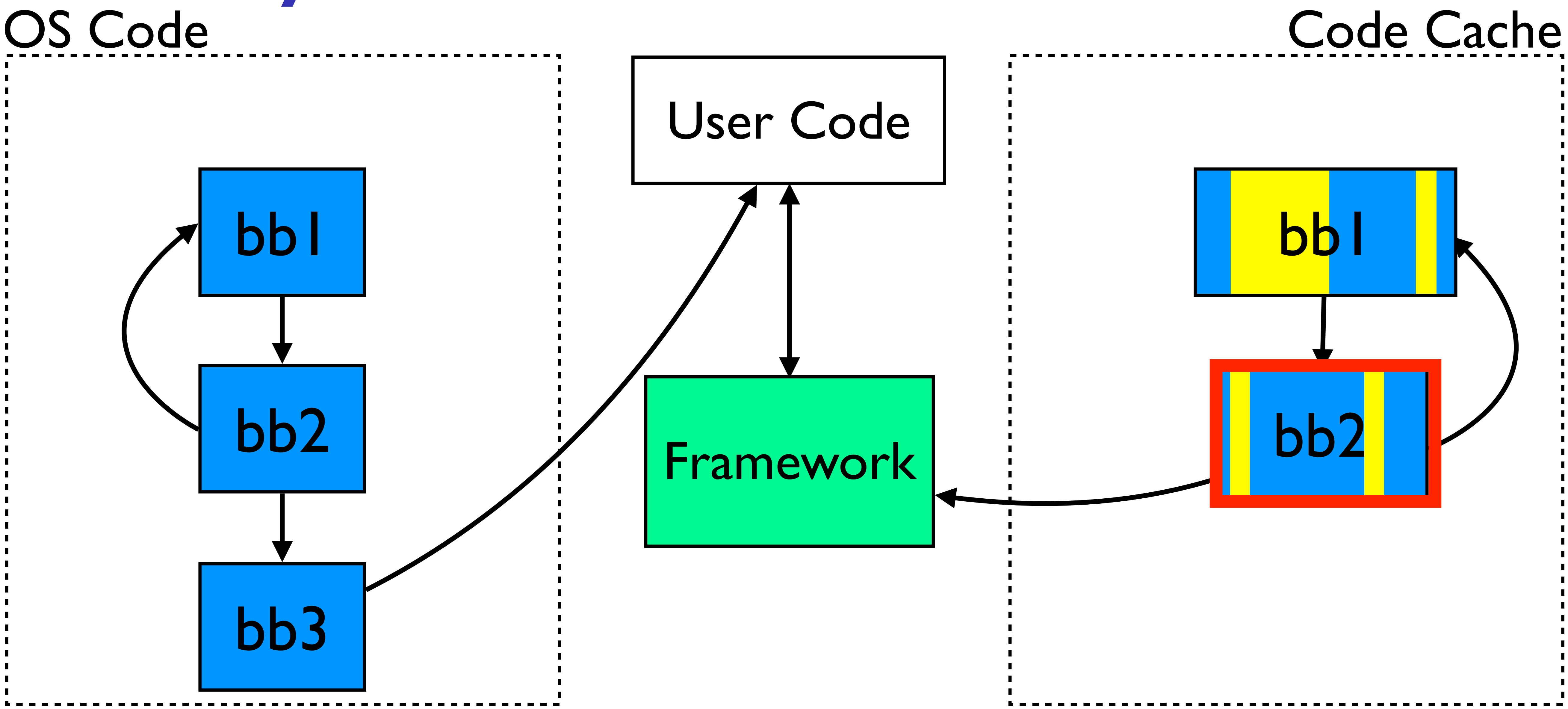
Dynamic Instrumentation



System call example

- bb1 is entry point

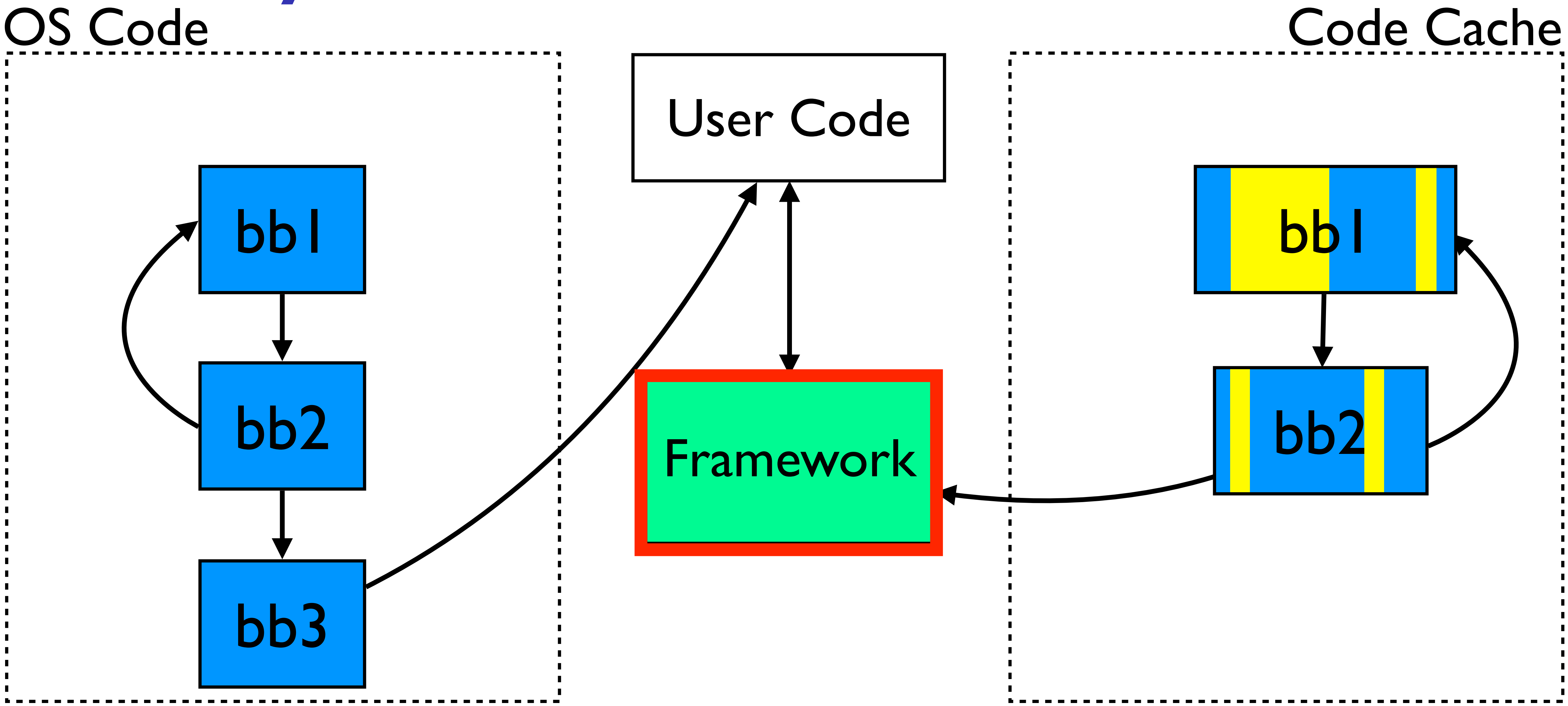
Dynamic Instrumentation



System call example

- bb1 is entry point

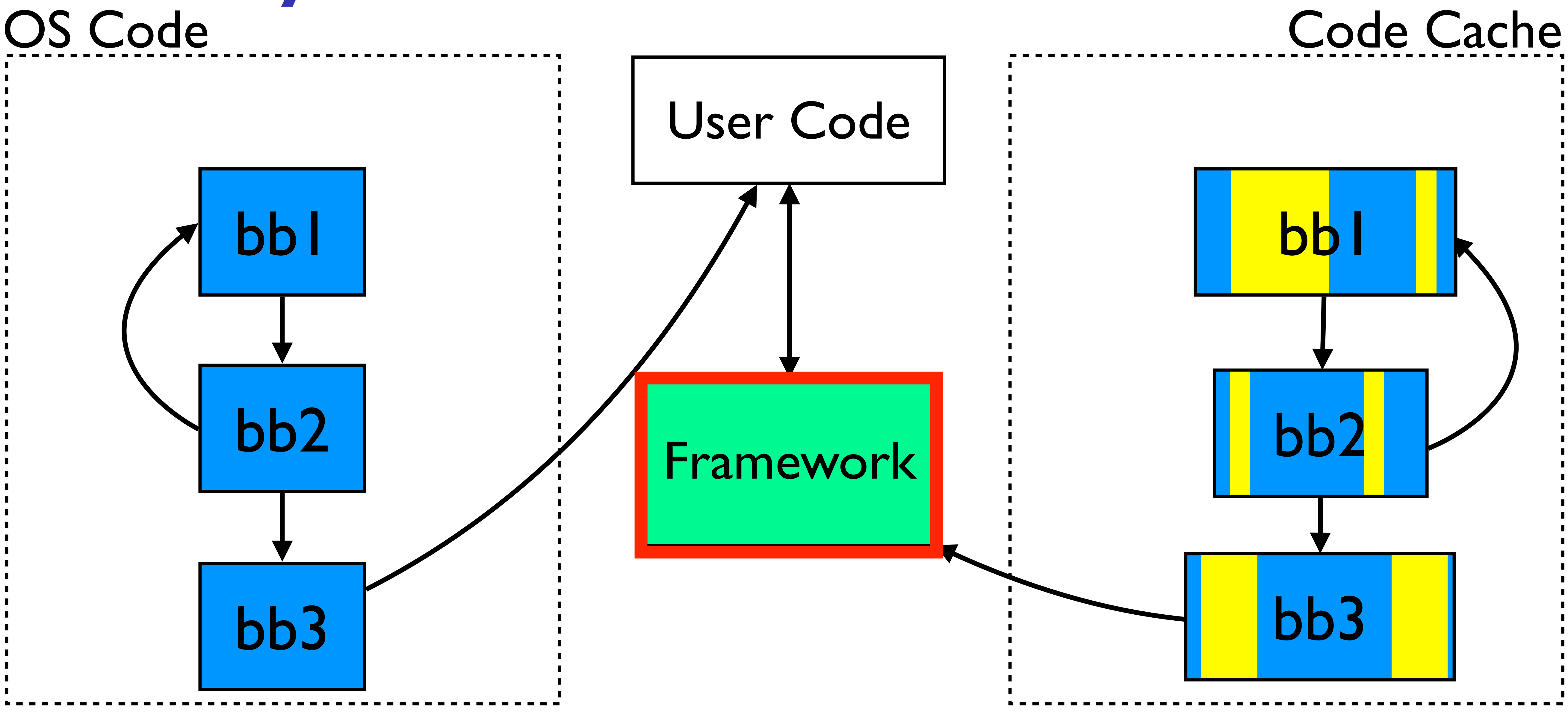
Dynamic Instrumentation



System call example

- bb1 is entry point

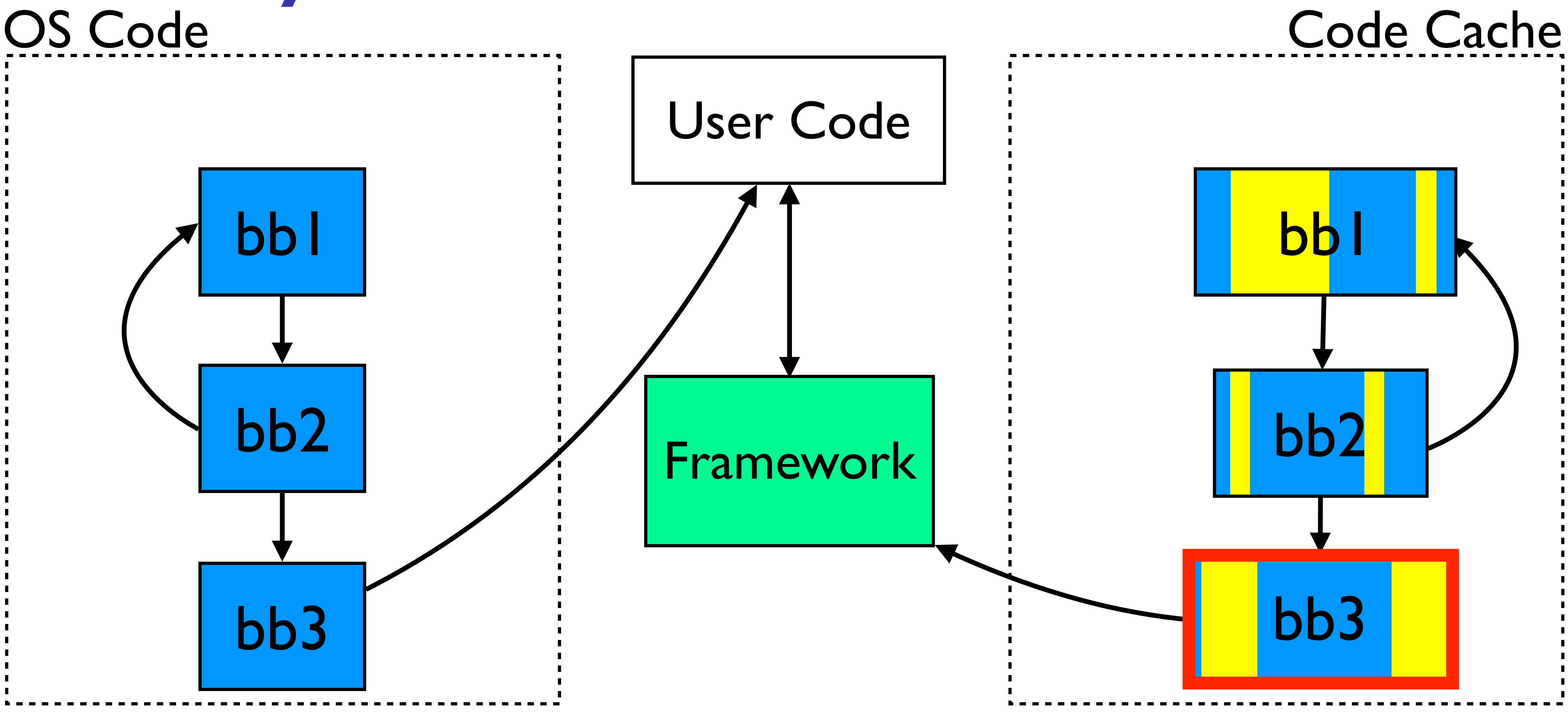
Dynamic Instrumentation



System call example

- bb1 is entry point

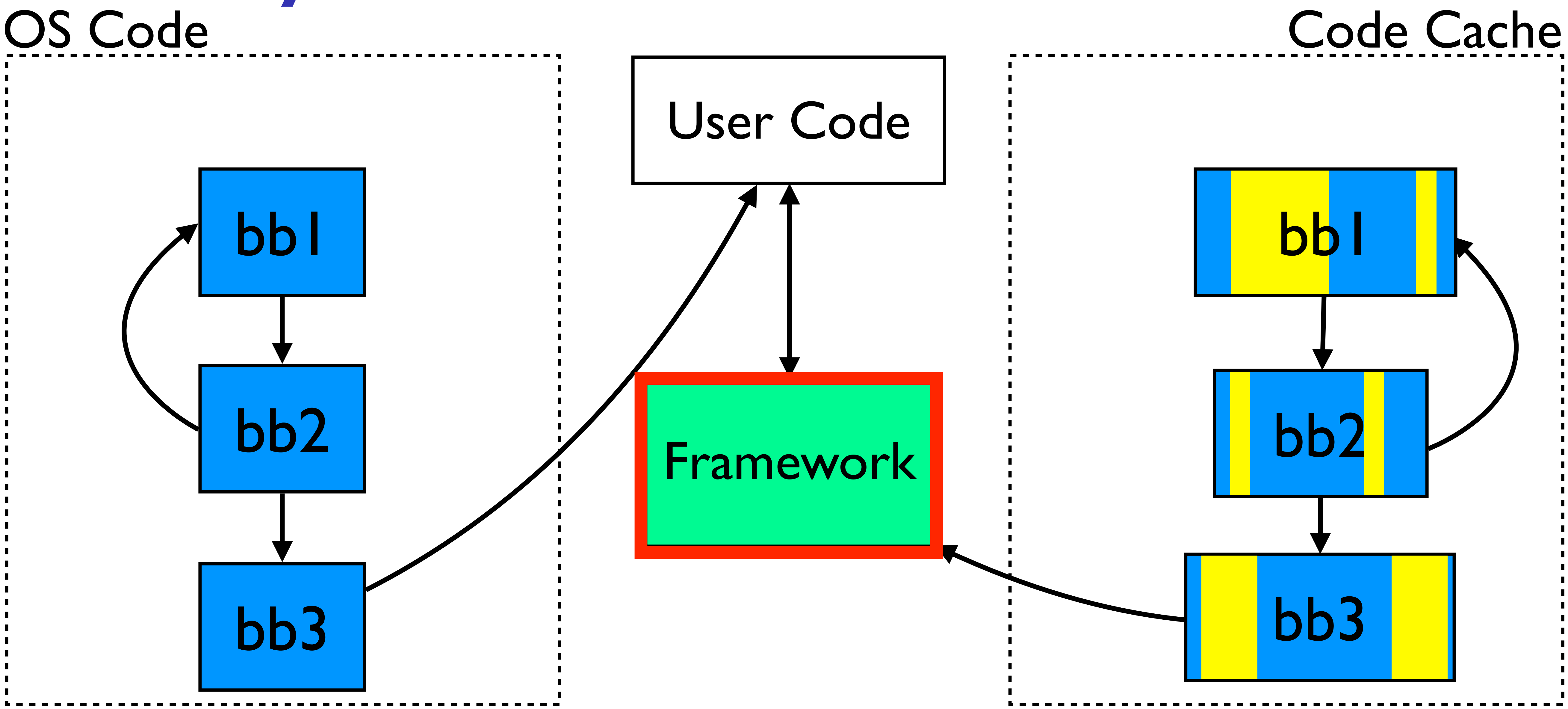
Dynamic Instrumentation



System call example

- bb1 is entry point

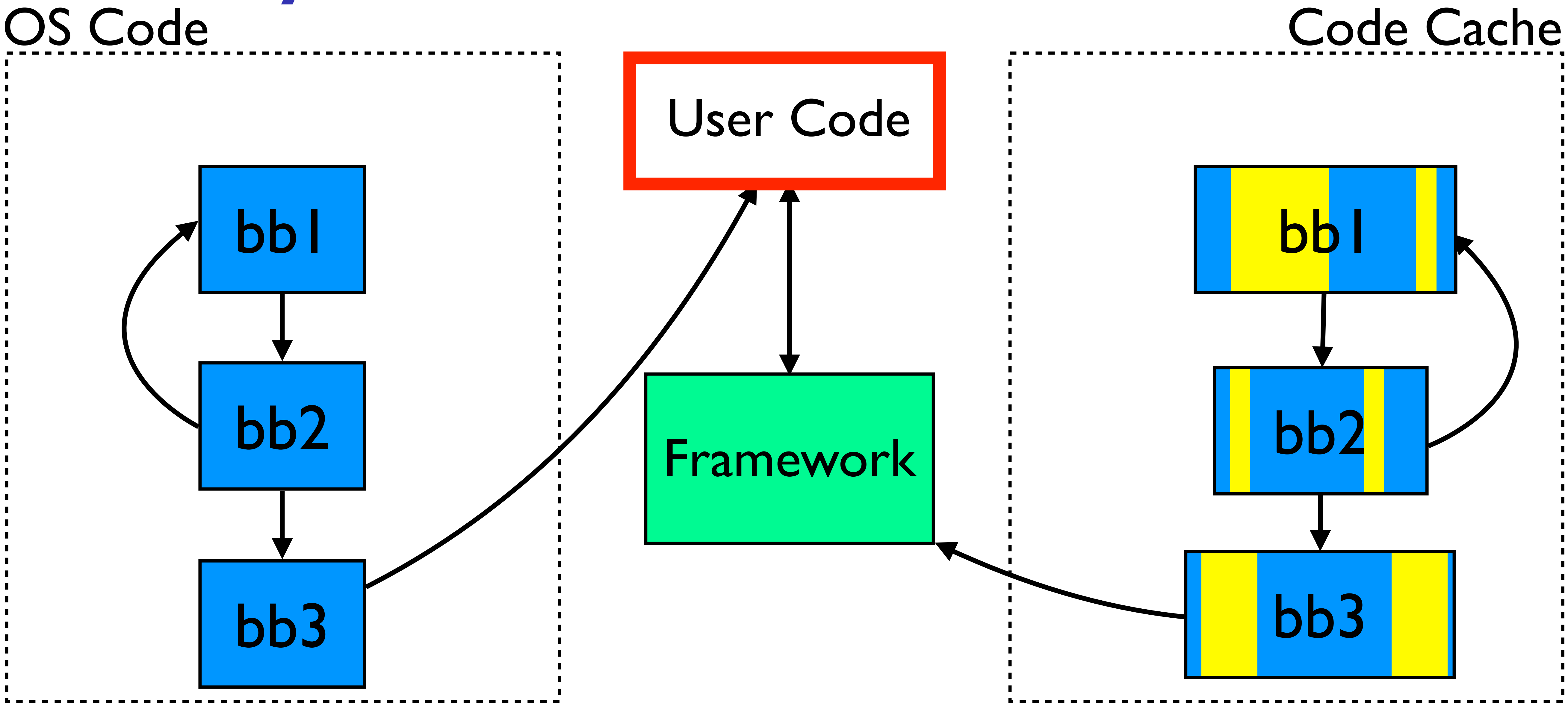
Dynamic Instrumentation



System call example

- bb1 is entry point

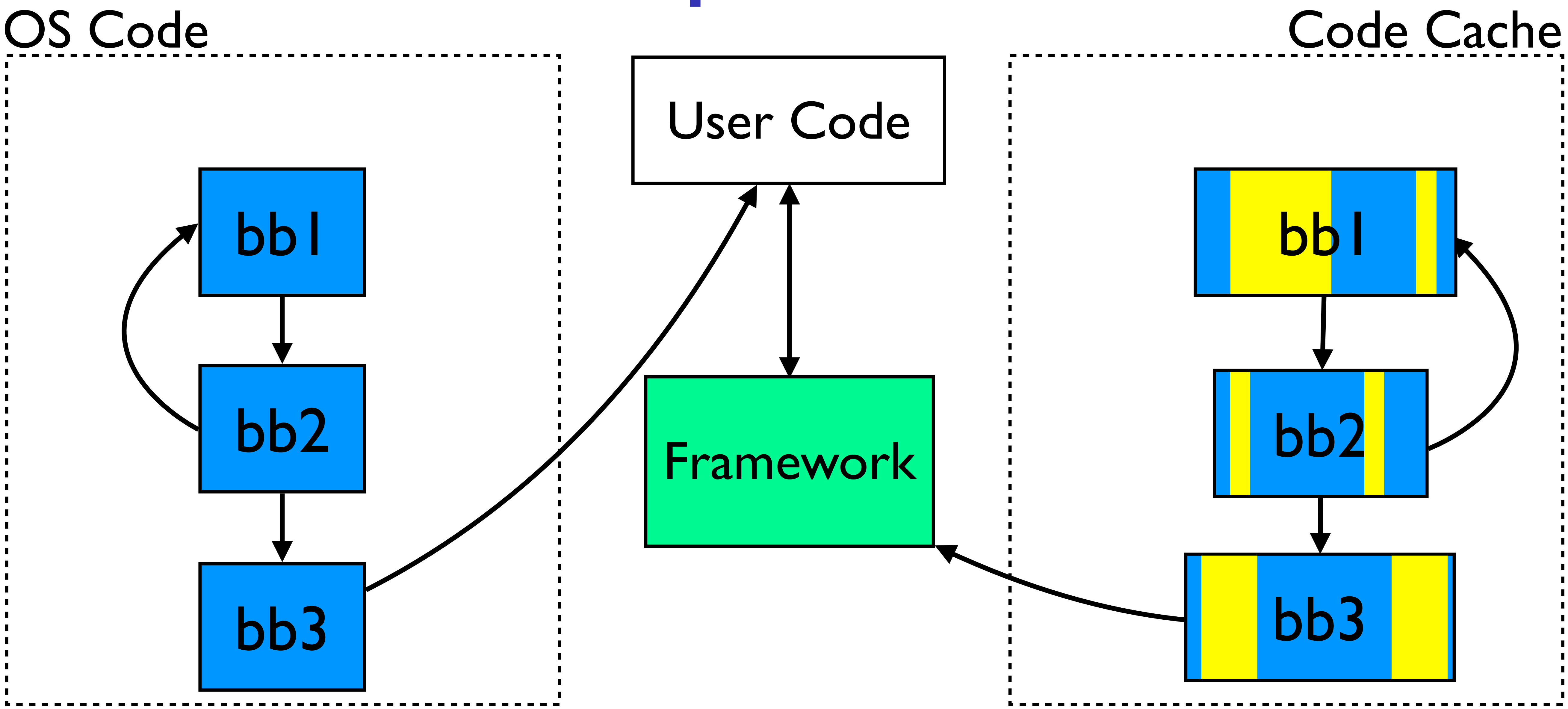
Dynamic Instrumentation



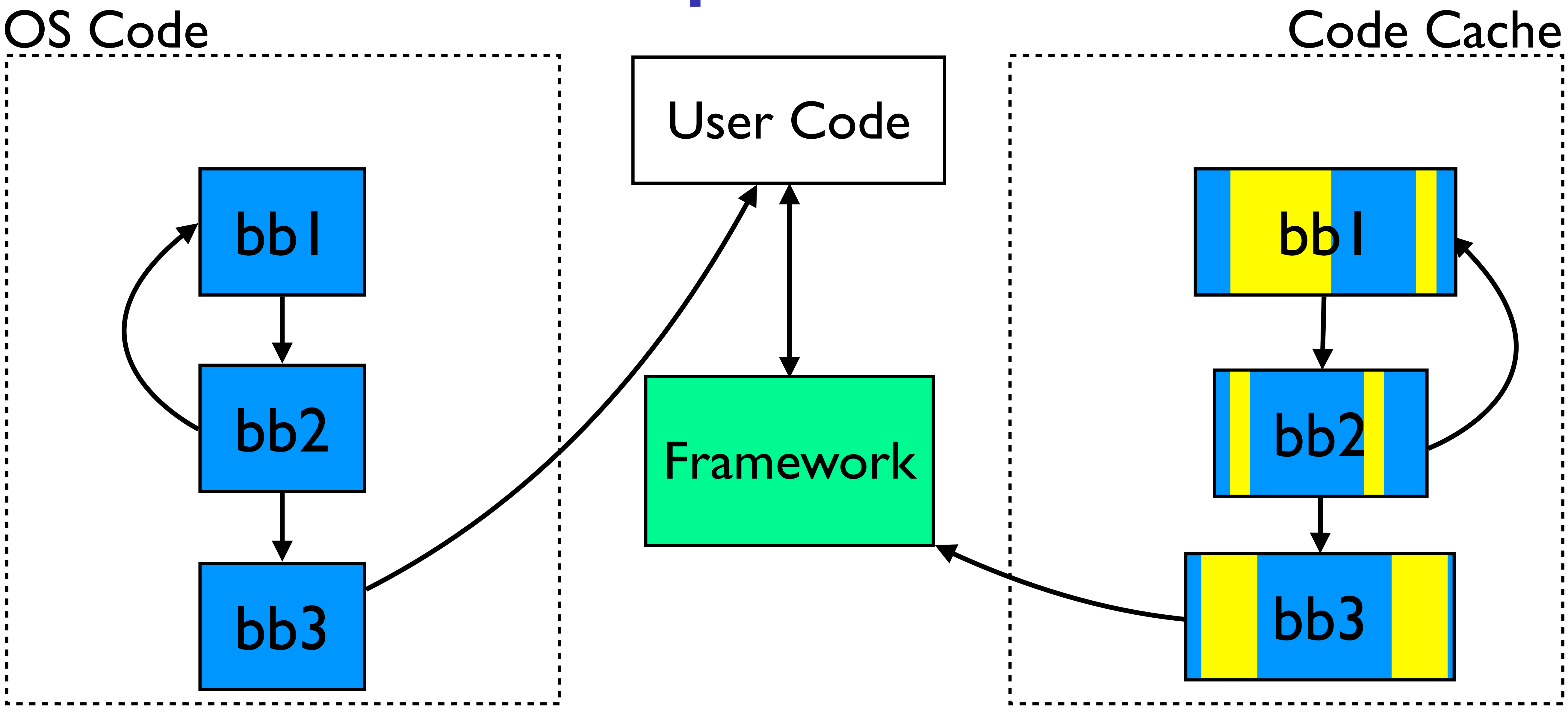
System call example

- bb1 is entry point

Complications



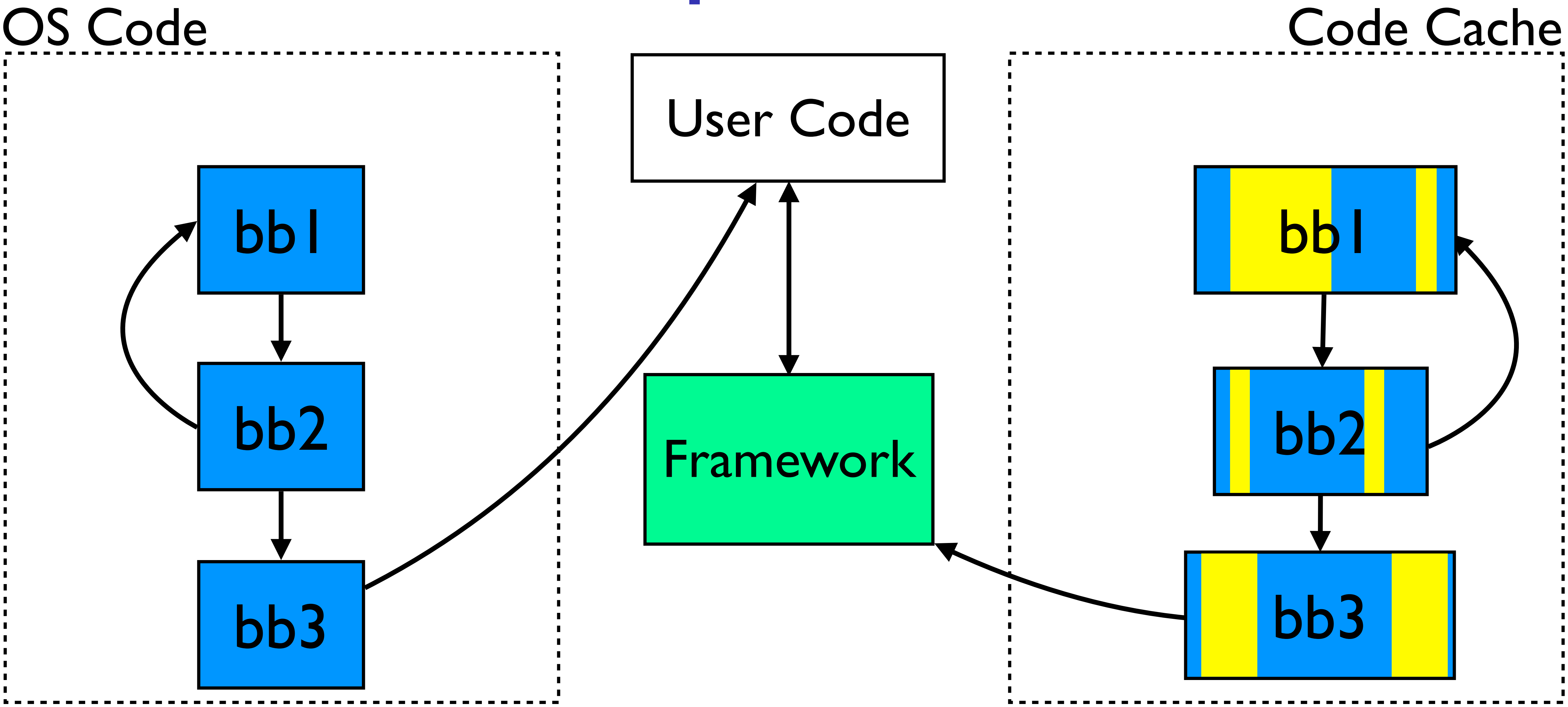
Complications



Reentrance

- How do you do I/O?
- Can't use OS

Complications



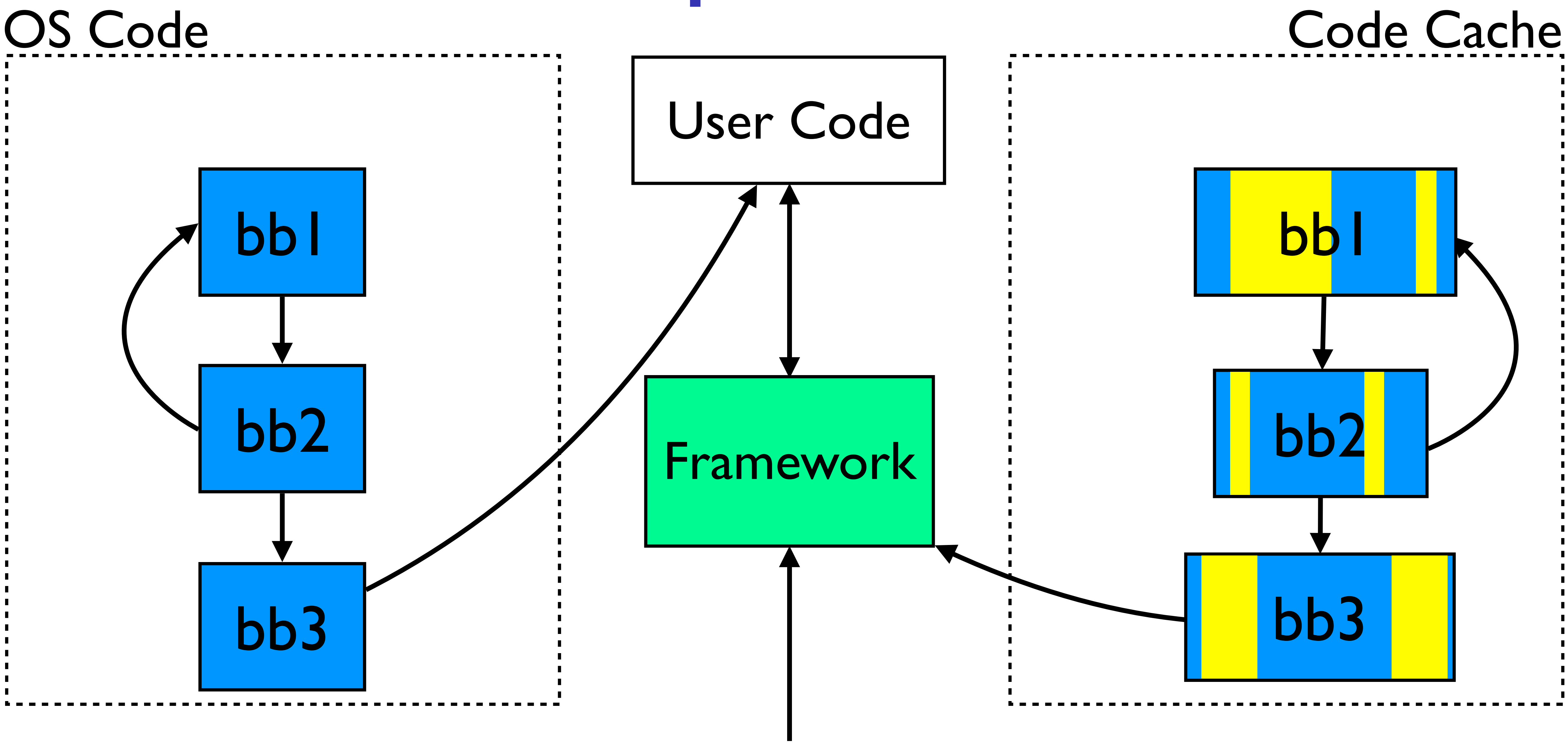
Reentrance

- How do you do I/O?
- Can't use OS

Concurrency

- Multiple CPUs using and building cache

Complications



Reentrance

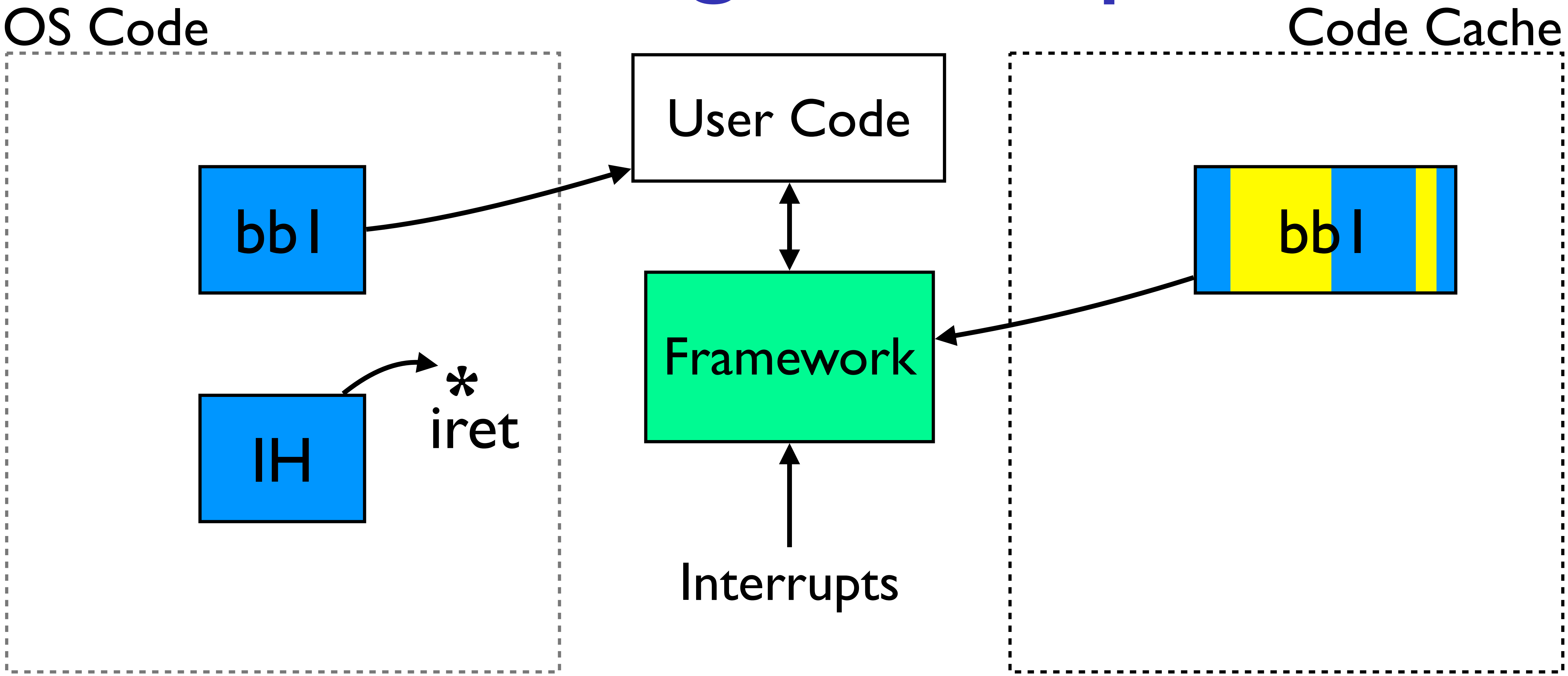
- How do you do I/O?
- Can’t use OS

Interrupts

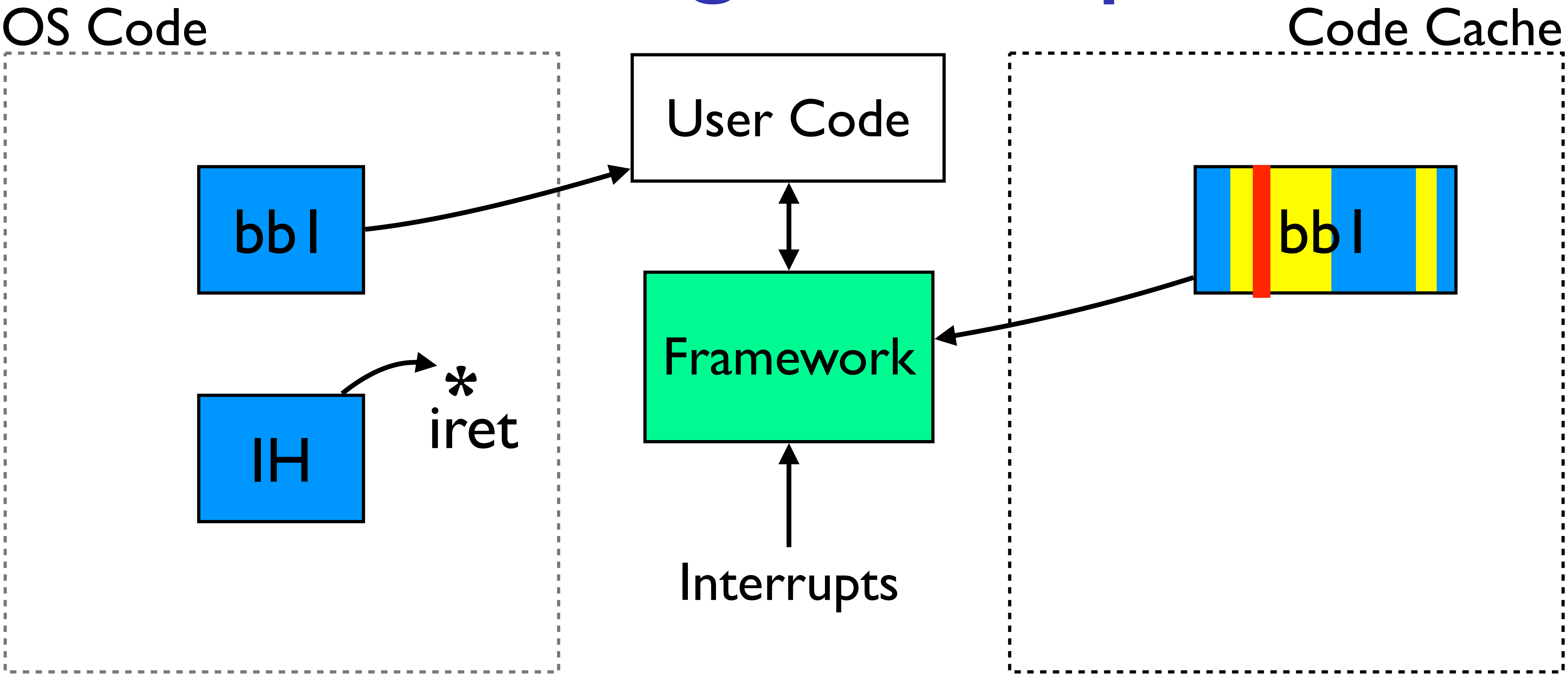
Concurrency

- Multiple CPUs using and building cache

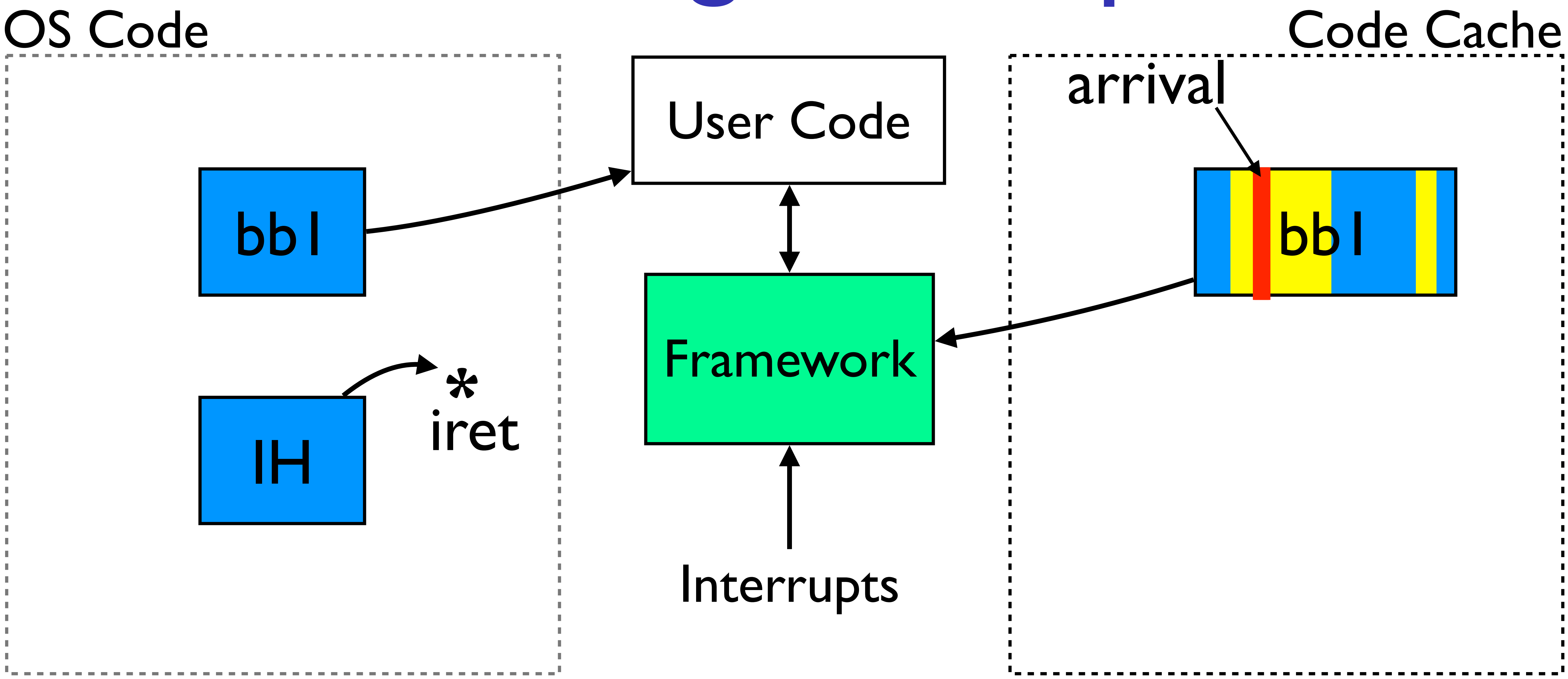
Handling Interrupts



Handling Interrupts



Handling Interrupts



OS Code

bb1

IH

*

iret

User Code

Framework

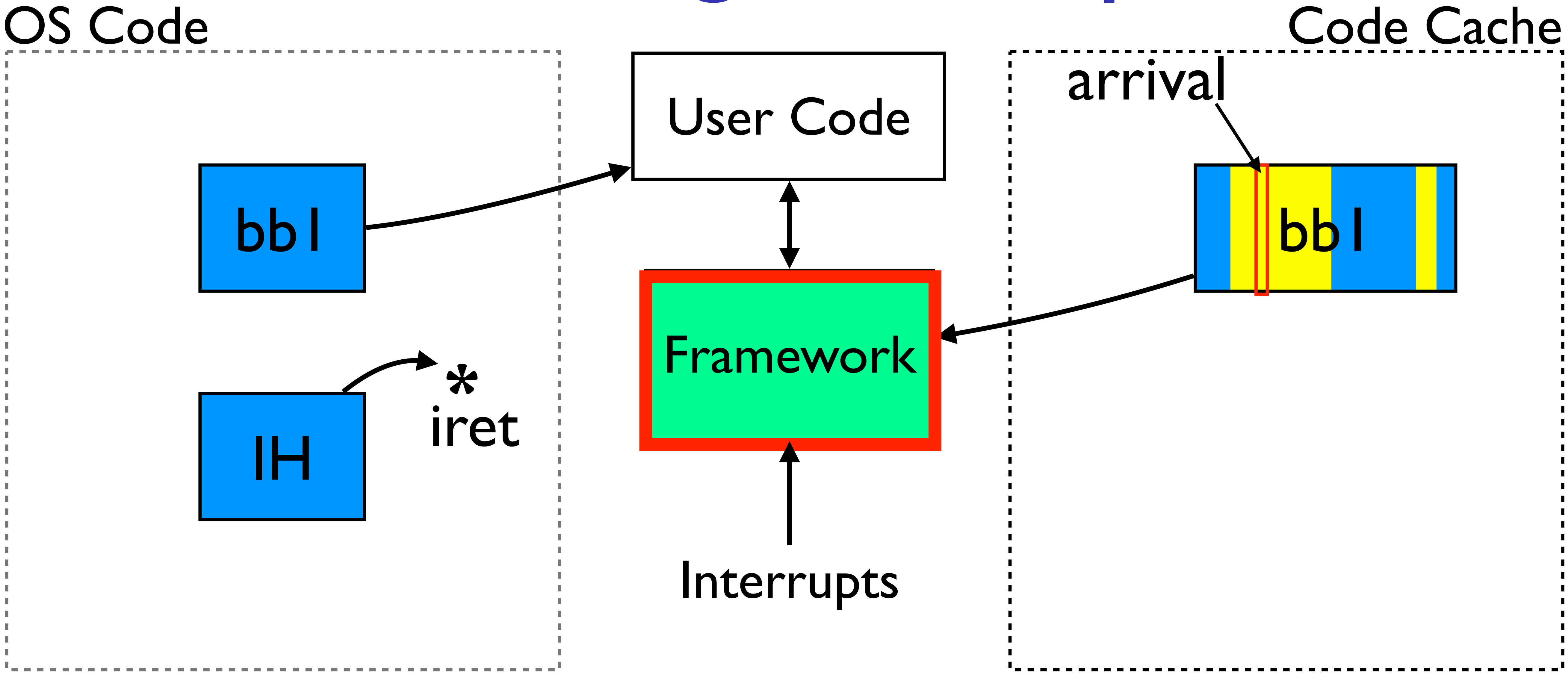
Interrupts

Code Cache

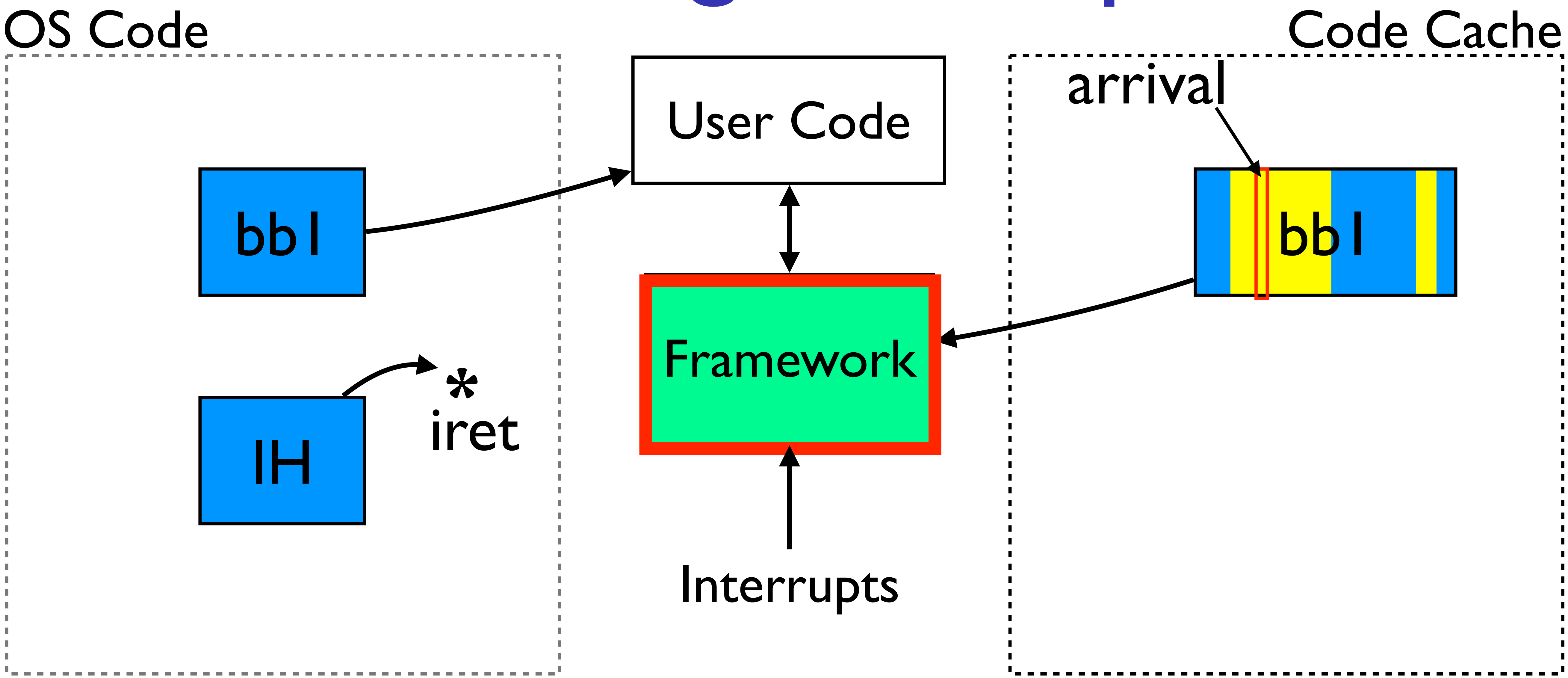
arrival

bb1

Handling Interrupts

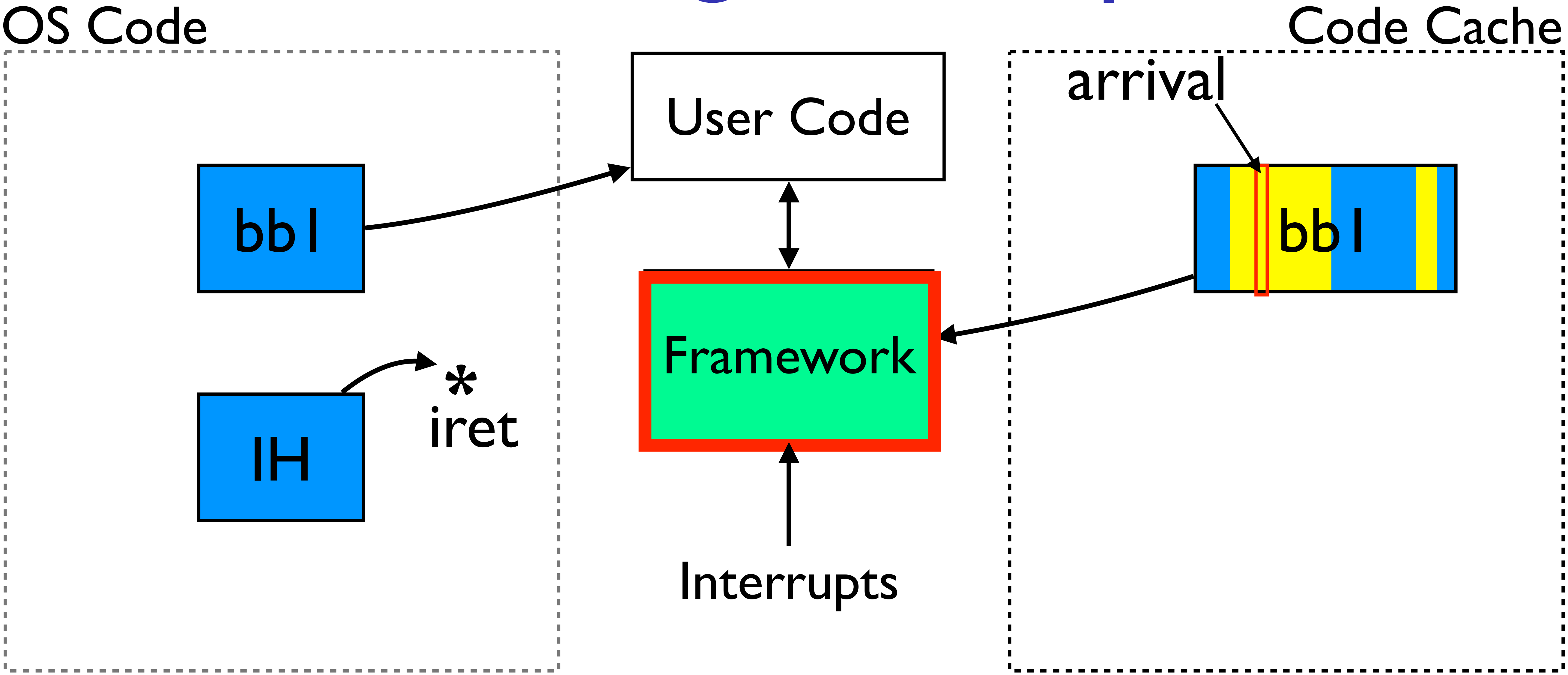


Handling Interrupts



What should framework do with interrupt?

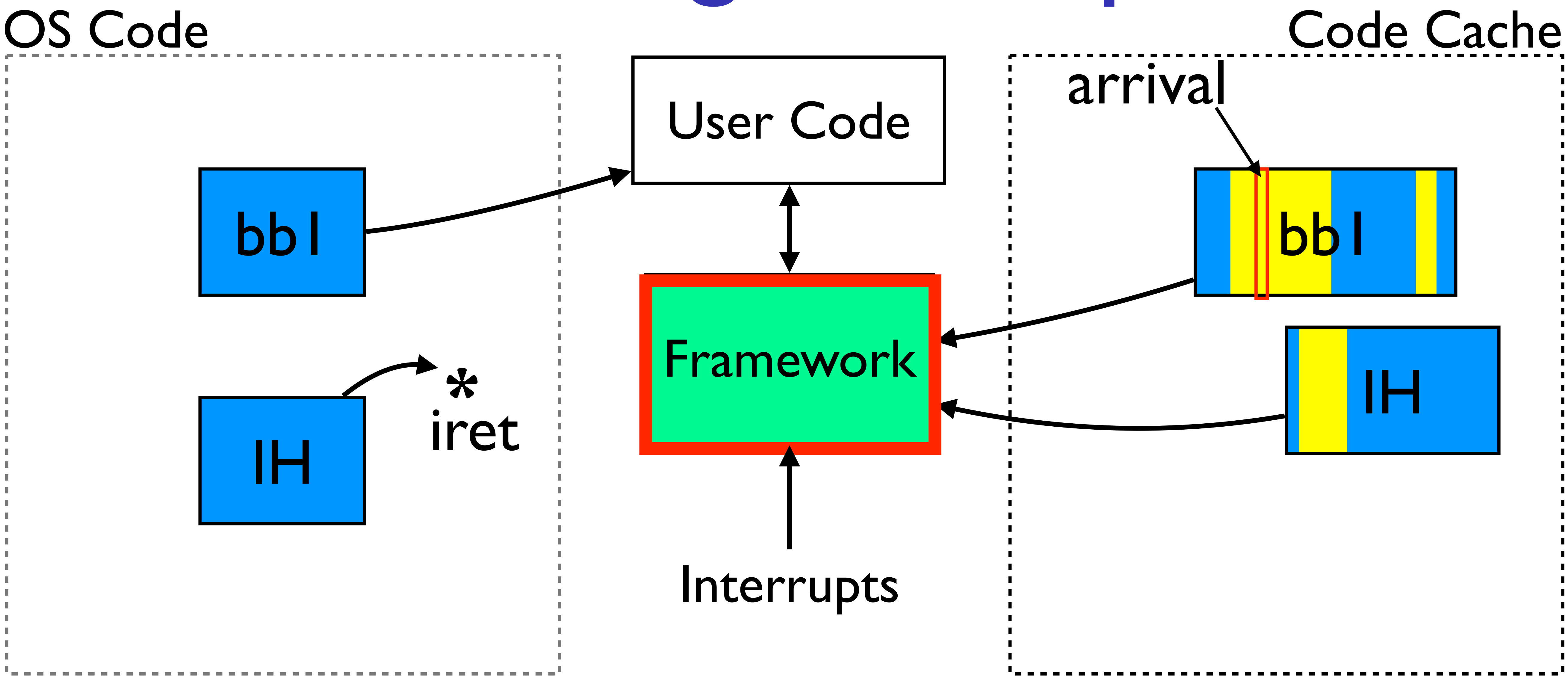
Handling Interrupts



What should framework do with interrupt?

- Can it run interrupt handler IH immediately?

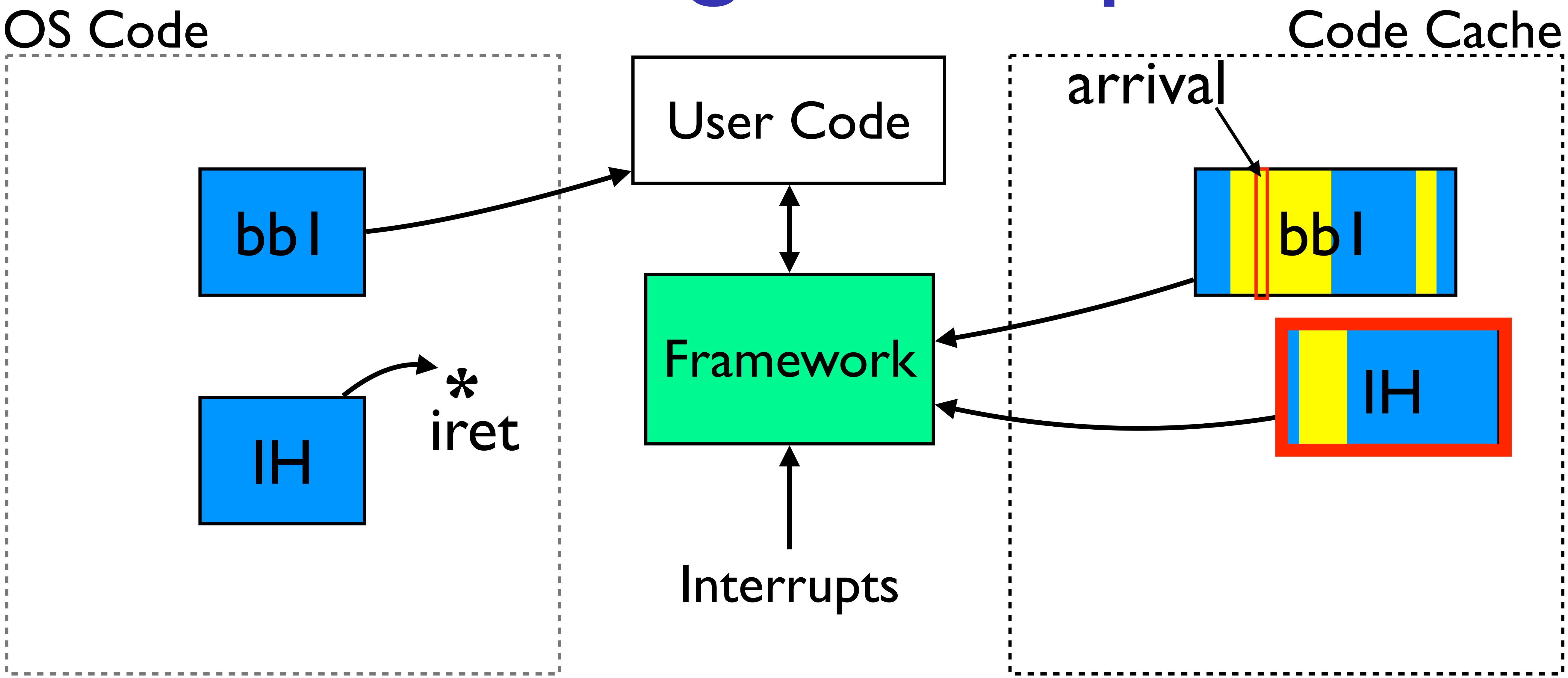
Handling Interrupts



What should framework do with interrupt?

- Can it run interrupt handler IH immediately?

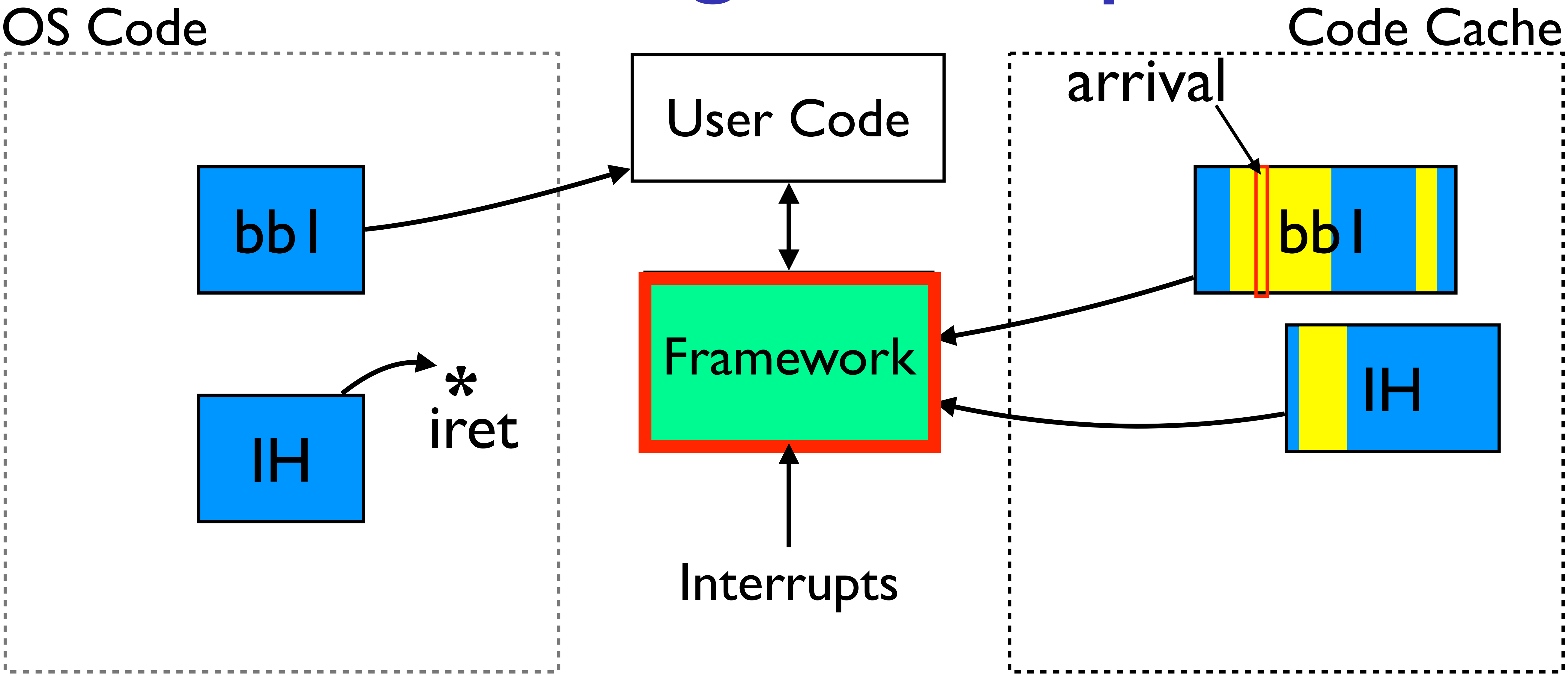
Handling Interrupts



What should framework do with interrupt?

- Can it run interrupt handler IH immediately?

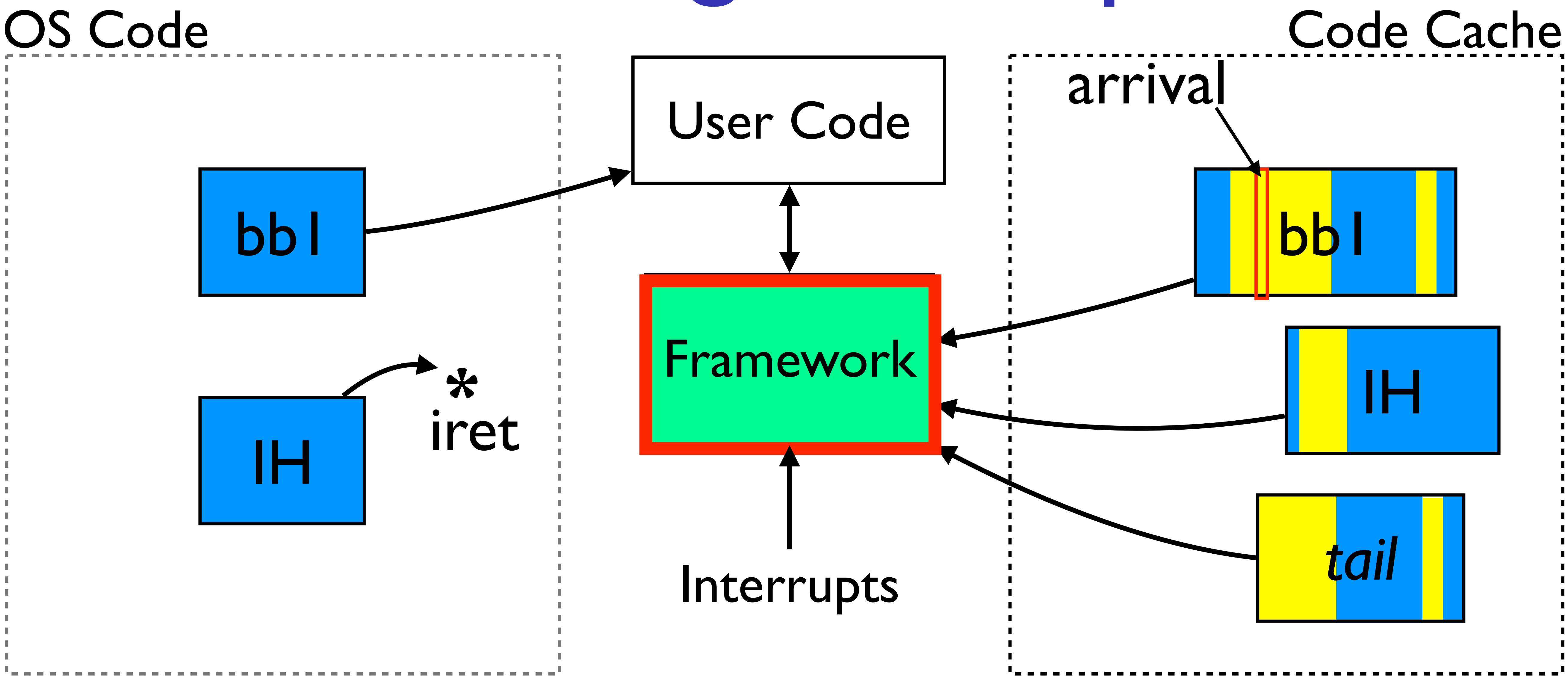
Handling Interrupts



What should framework do with interrupt?

- Can it run interrupt handler IH immediately?

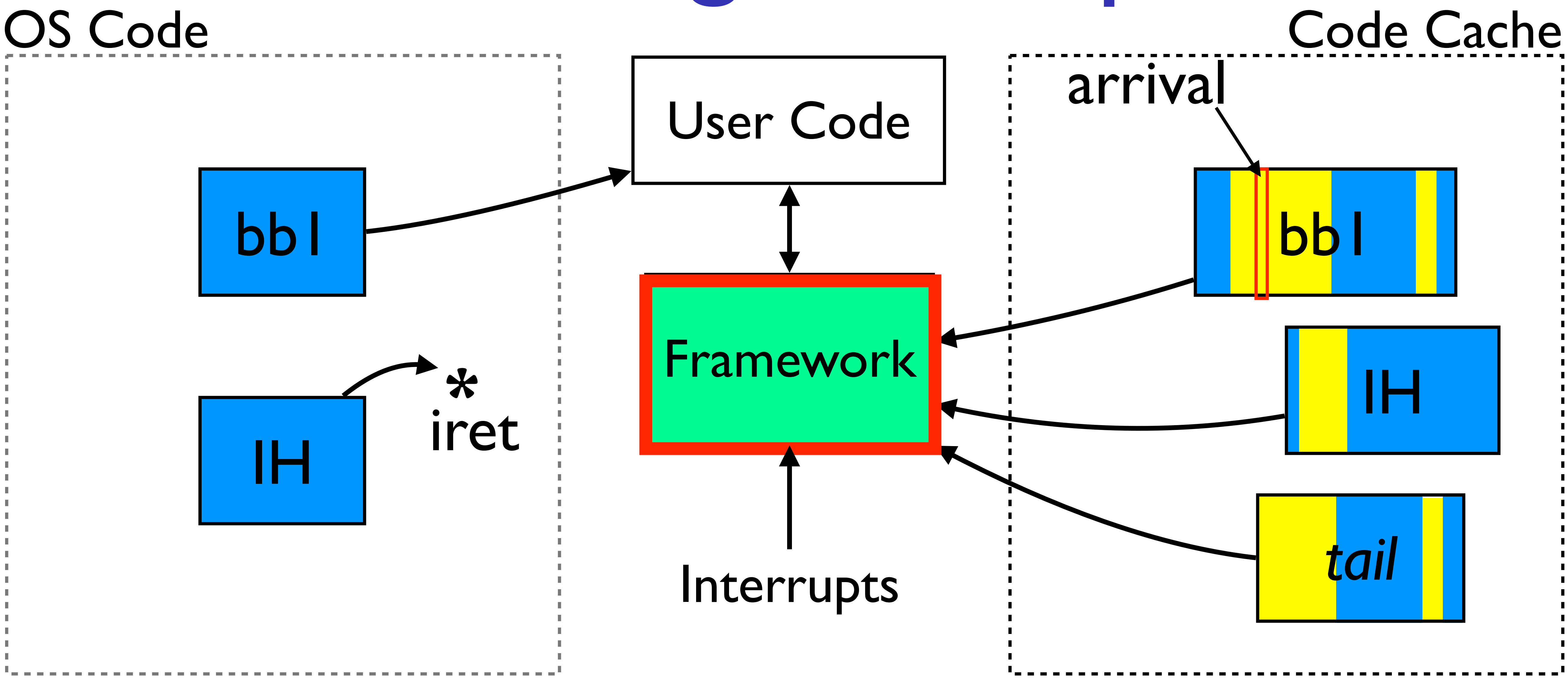
Handling Interrupts



What should framework do with interrupt?

- Can it run interrupt handler IH immediately?

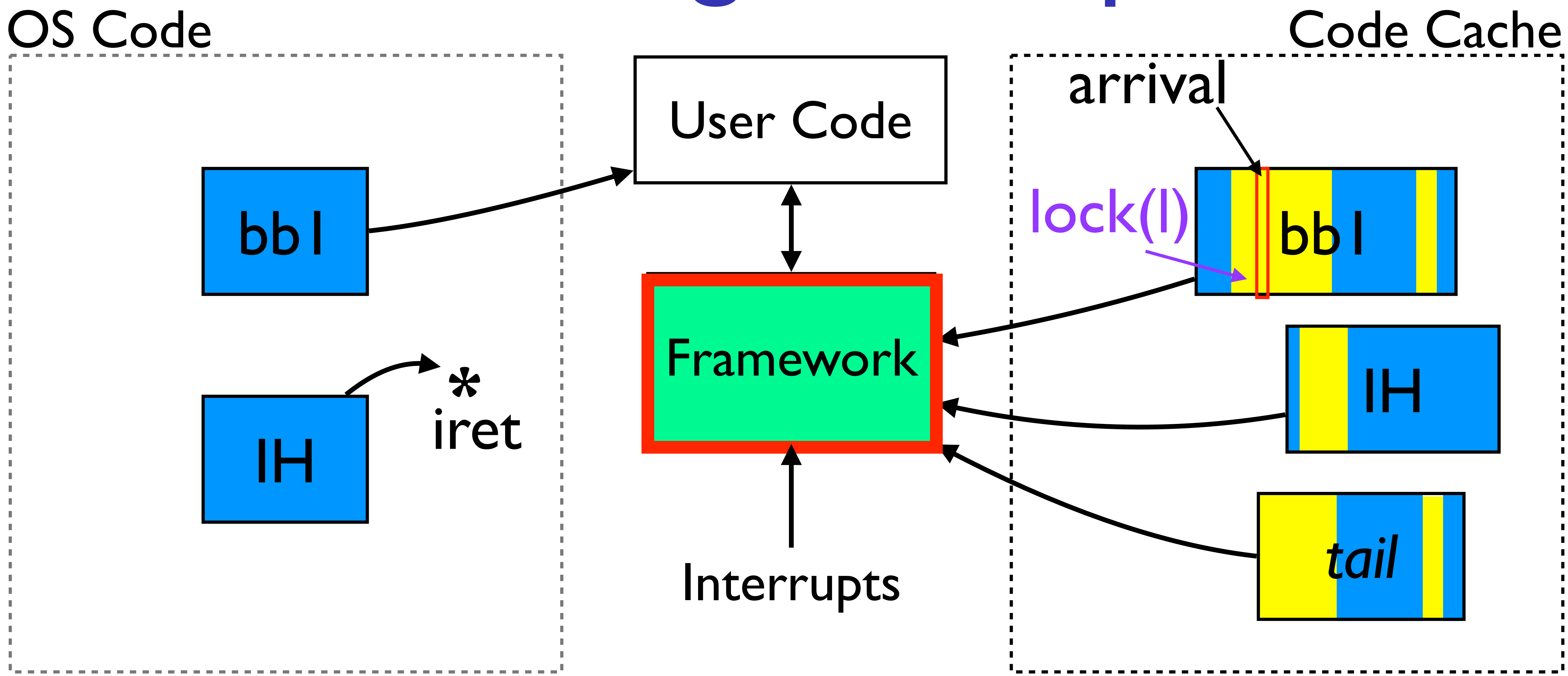
Handling Interrupts



What should framework do with interrupt?

- Can it run interrupt handler IH immediately?
  - Problem if instrumentation isn't reentrant

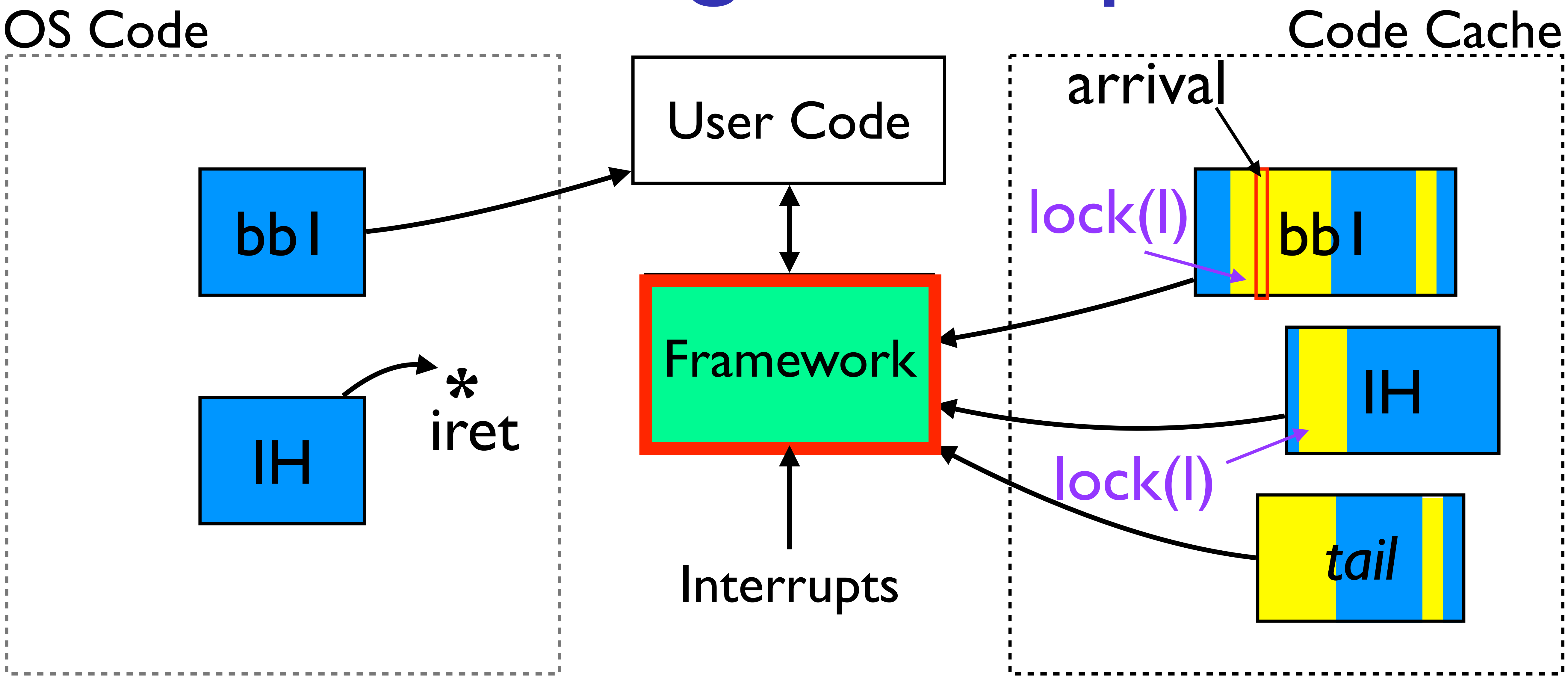
Handling Interrupts



What should framework do with interrupt?

- Can it run interrupt handler IH immediately?
  - Problem if instrumentation isn't reentrant

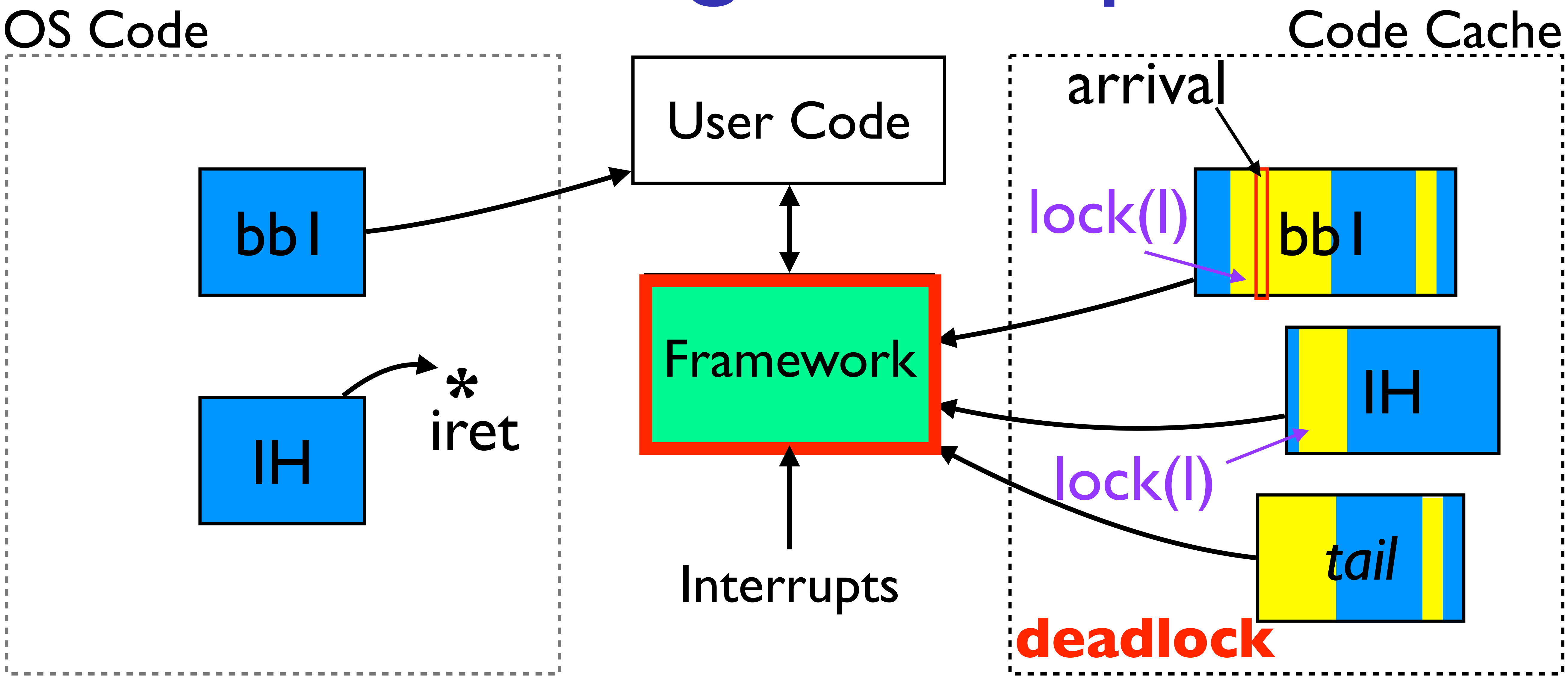
Handling Interrupts



What should framework do with interrupt?

- Can it run interrupt handler IH immediately?
  - Problem if instrumentation isn't reentrant

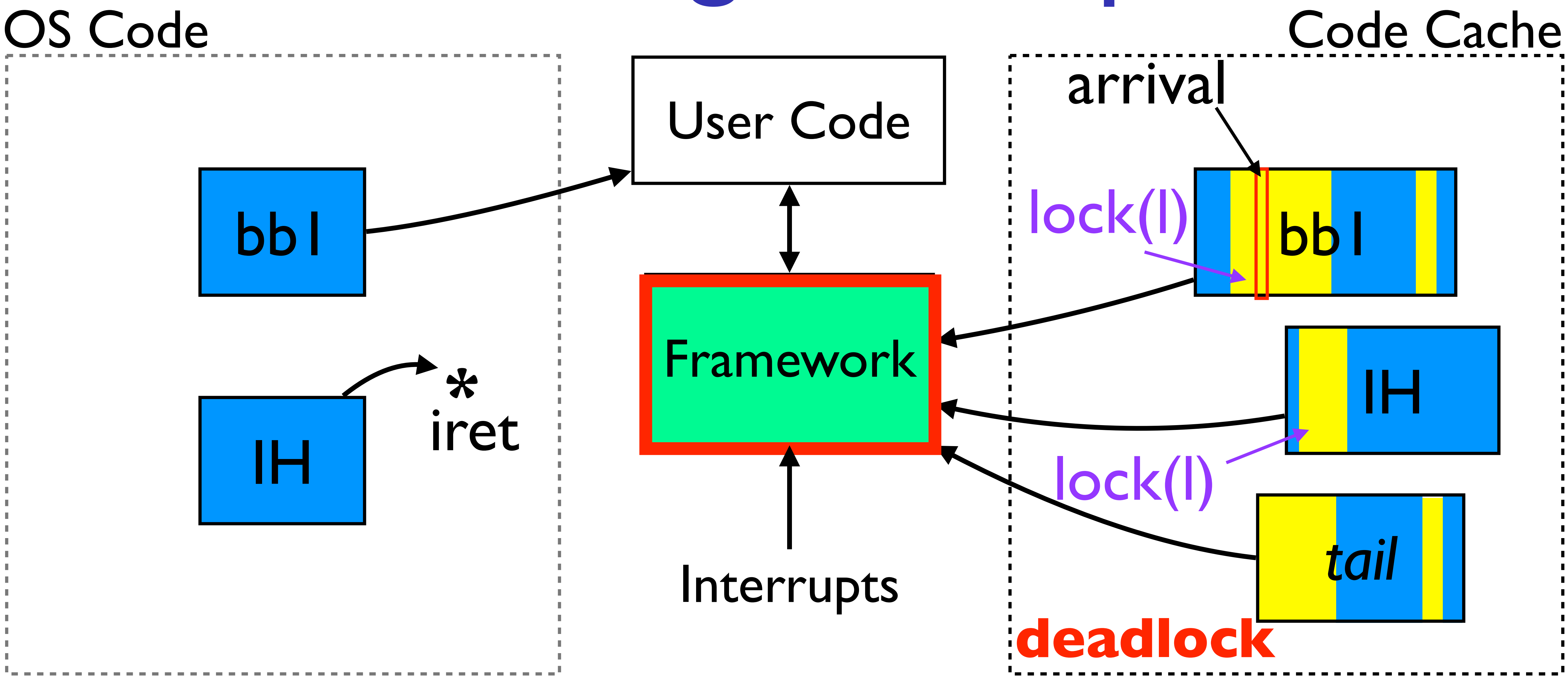
Handling Interrupts



What should framework do with interrupt?

- Can it run interrupt handler IH immediately?
    - Problem if instrumentation isn't reentrant

Handling Interrupts

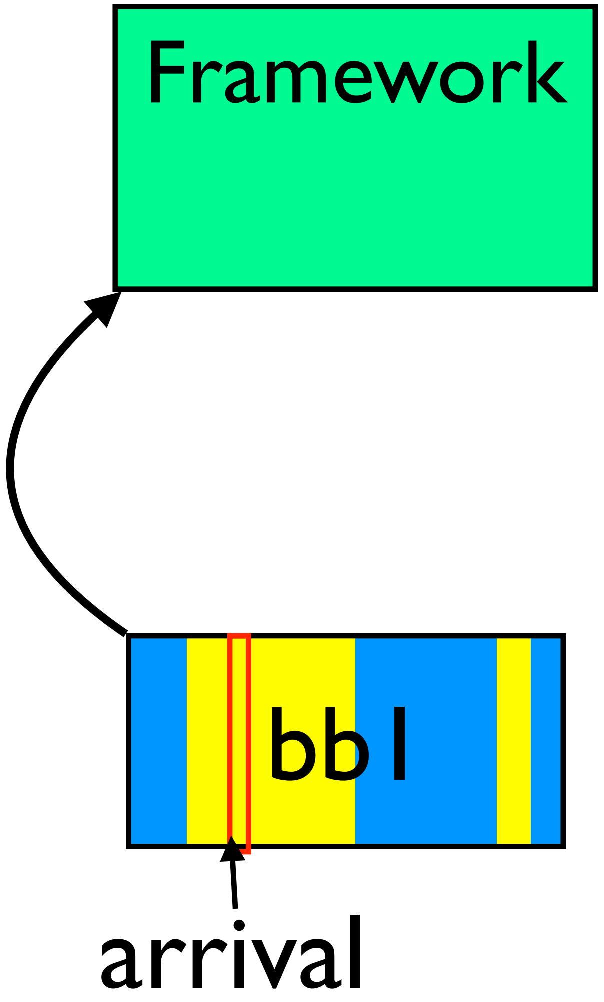


What should framework do with interrupt?

- Can it run interrupt handler IH immediately?
  - Problem if instrumentation isn't reentrant

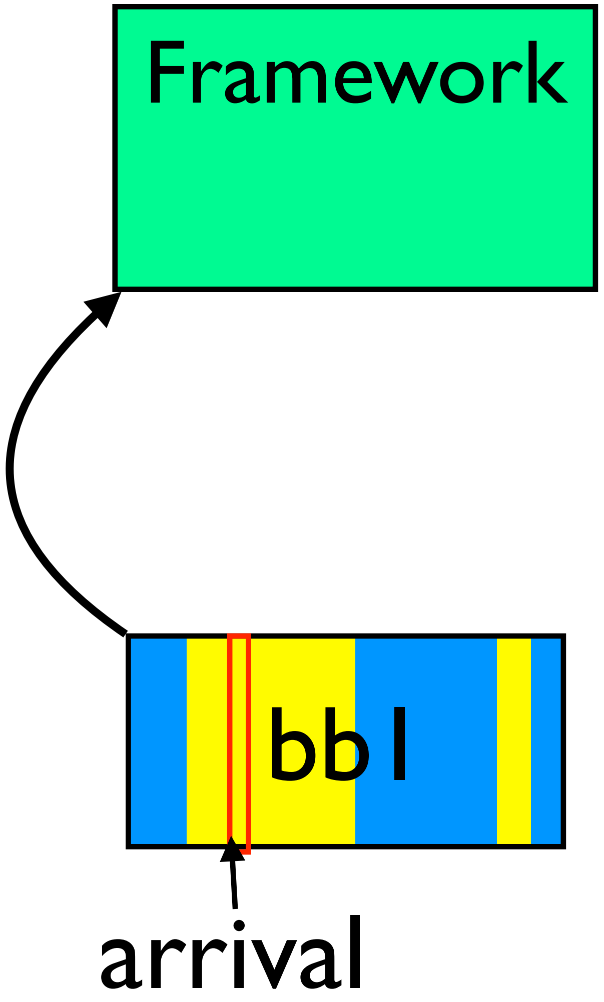
Need to delay

Delaying Interrupts



Delaying Interrupts

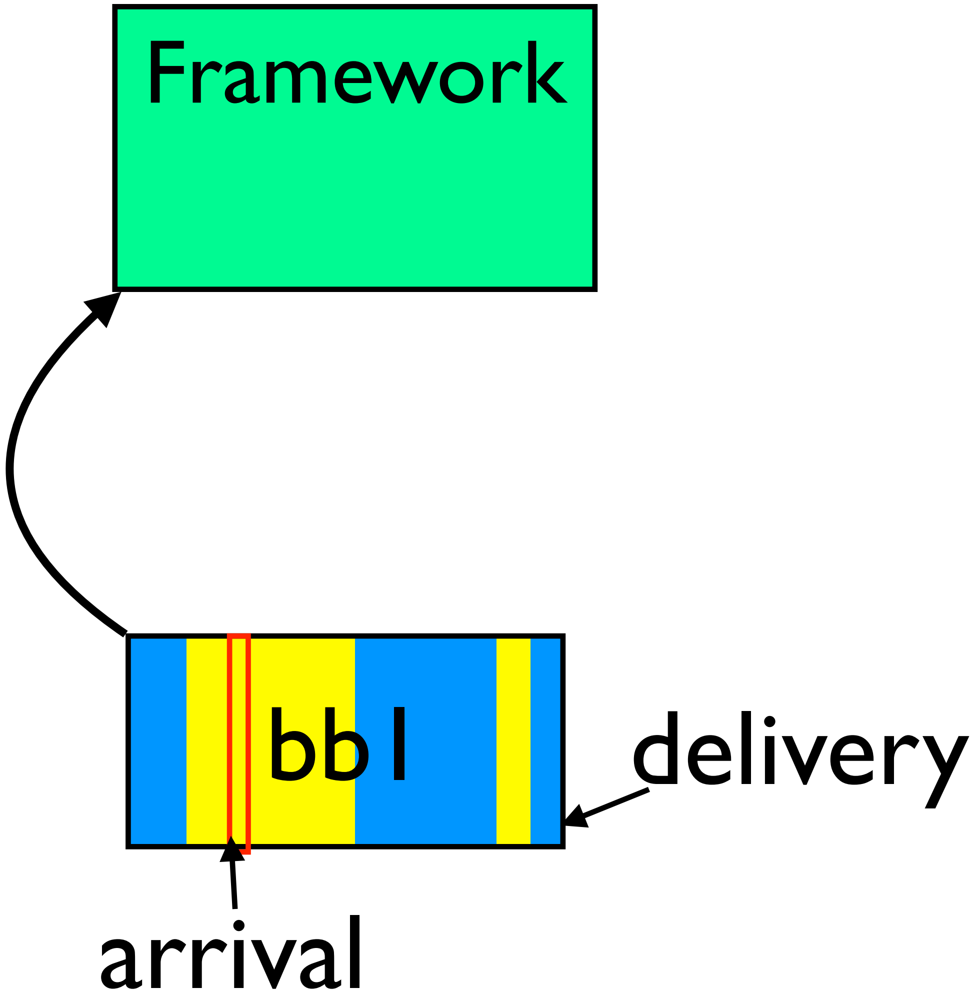
Where do we delay it until?



Delaying Interrupts

Where do we delay it until?

Delay until end of bb1?

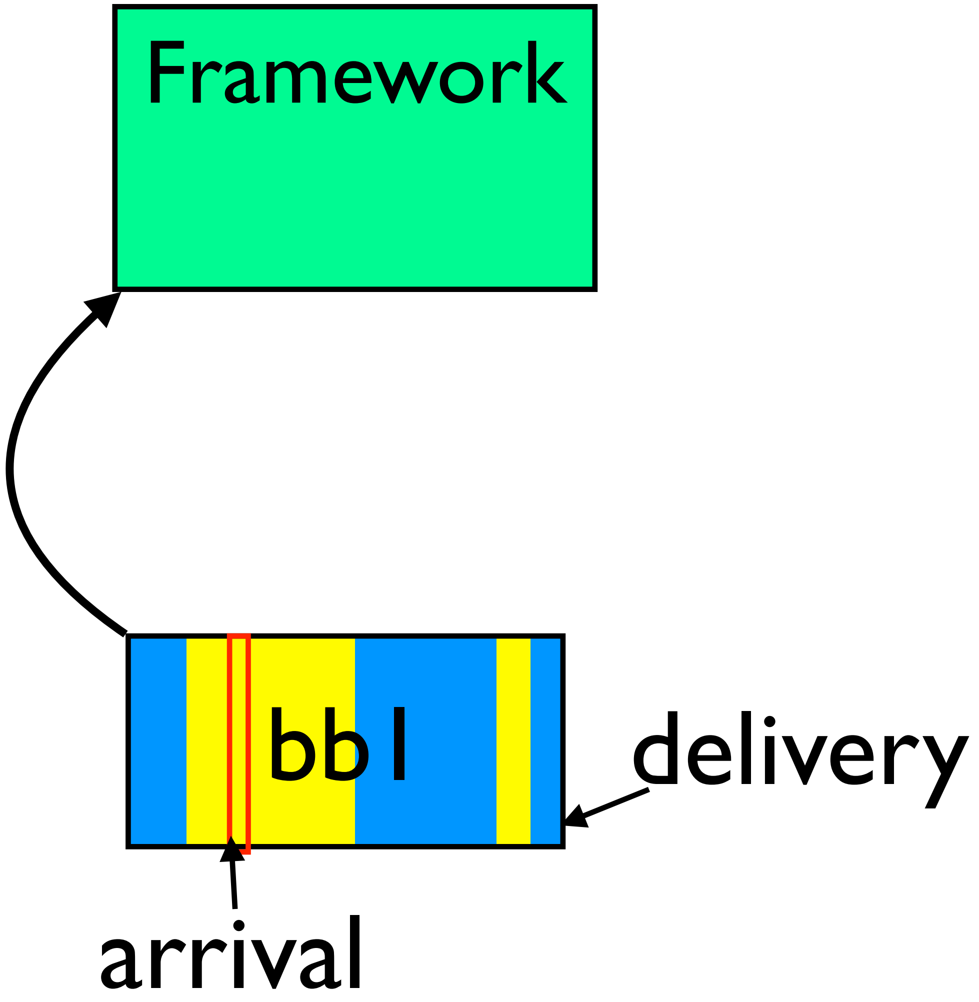


Delaying Interrupts

Where do we delay it until?

Delay until end of bb1?

- Avoids tail

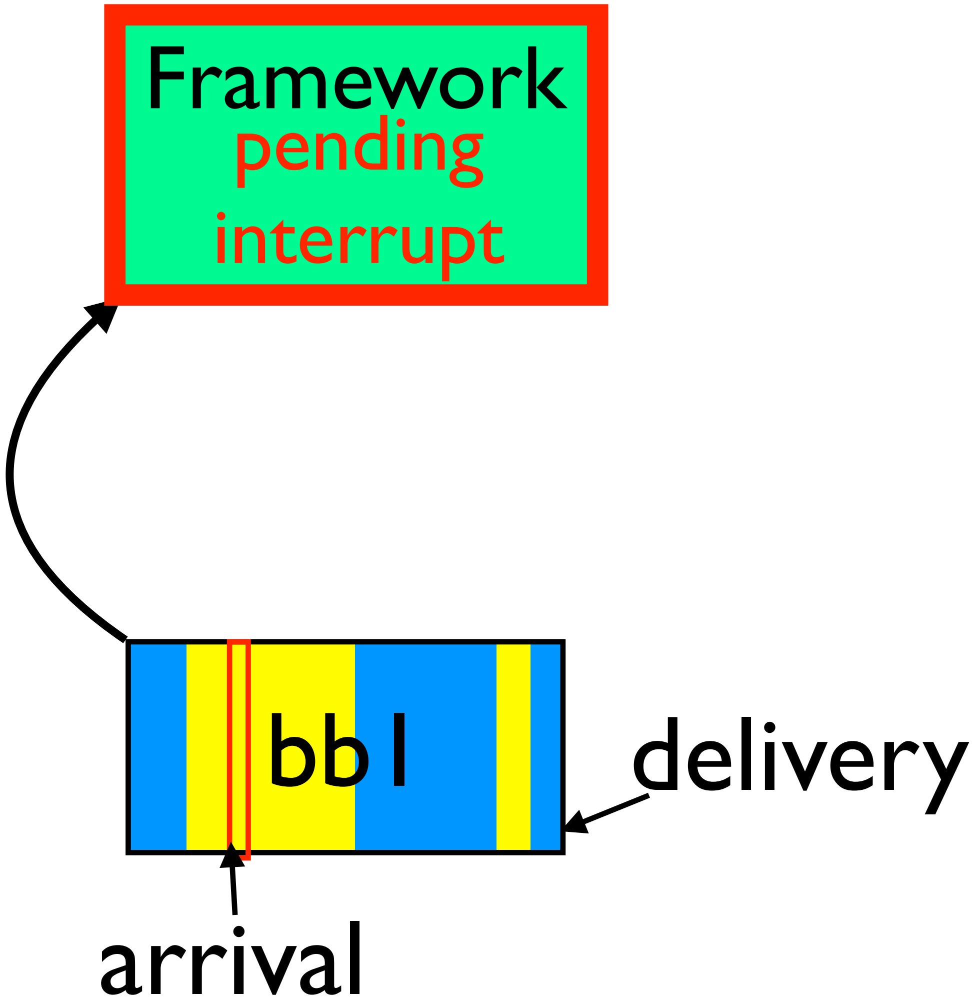


Delaying Interrupts

Where do we delay it until?

Delay until end of bb1?

- Avoids tail

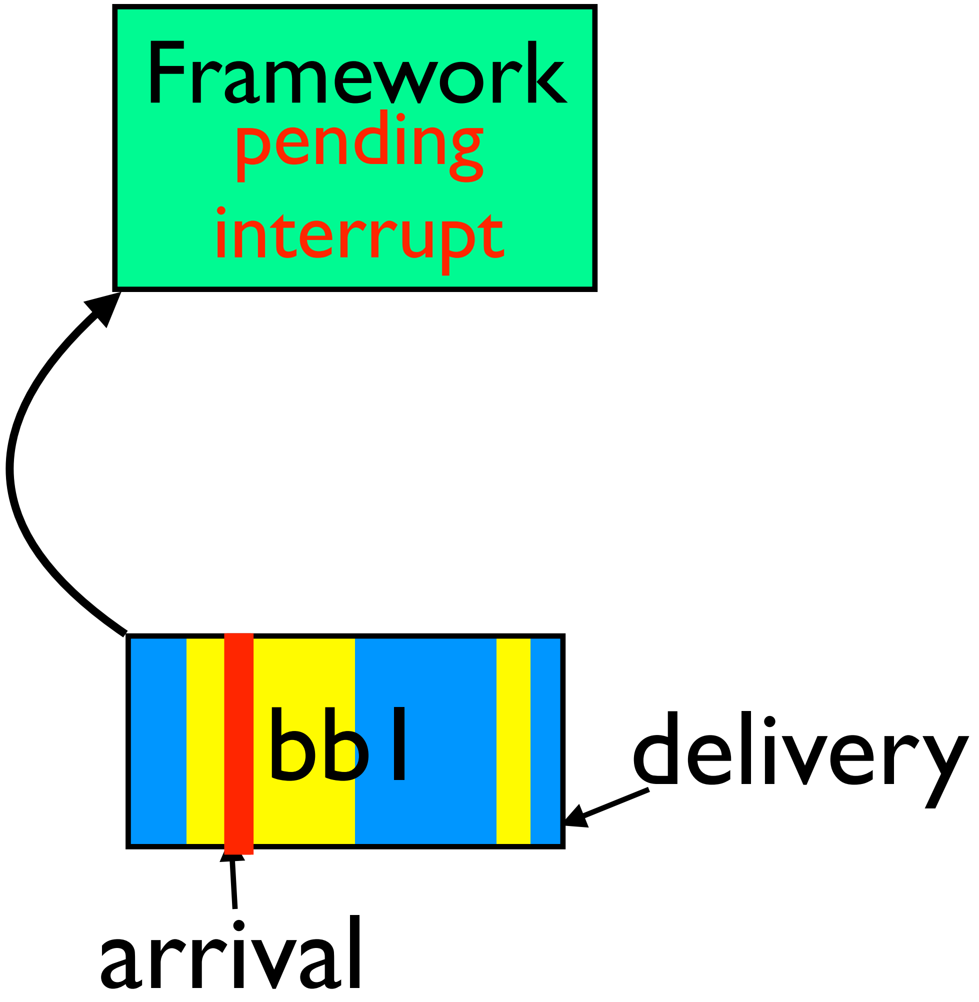


Delaying Interrupts

Where do we delay it until?

Delay until end of bb1?

- Avoids tail

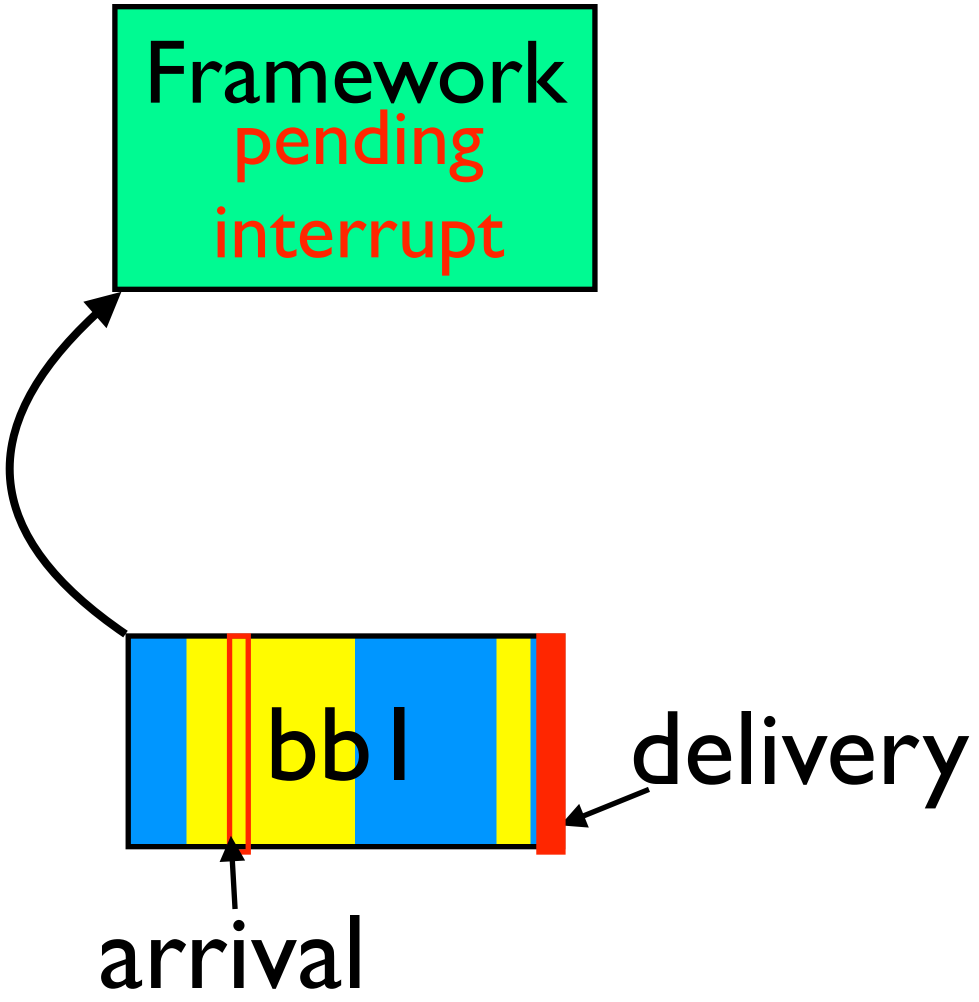


Delaying Interrupts

Where do we delay it until?

Delay until end of bb1?

- Avoids tail

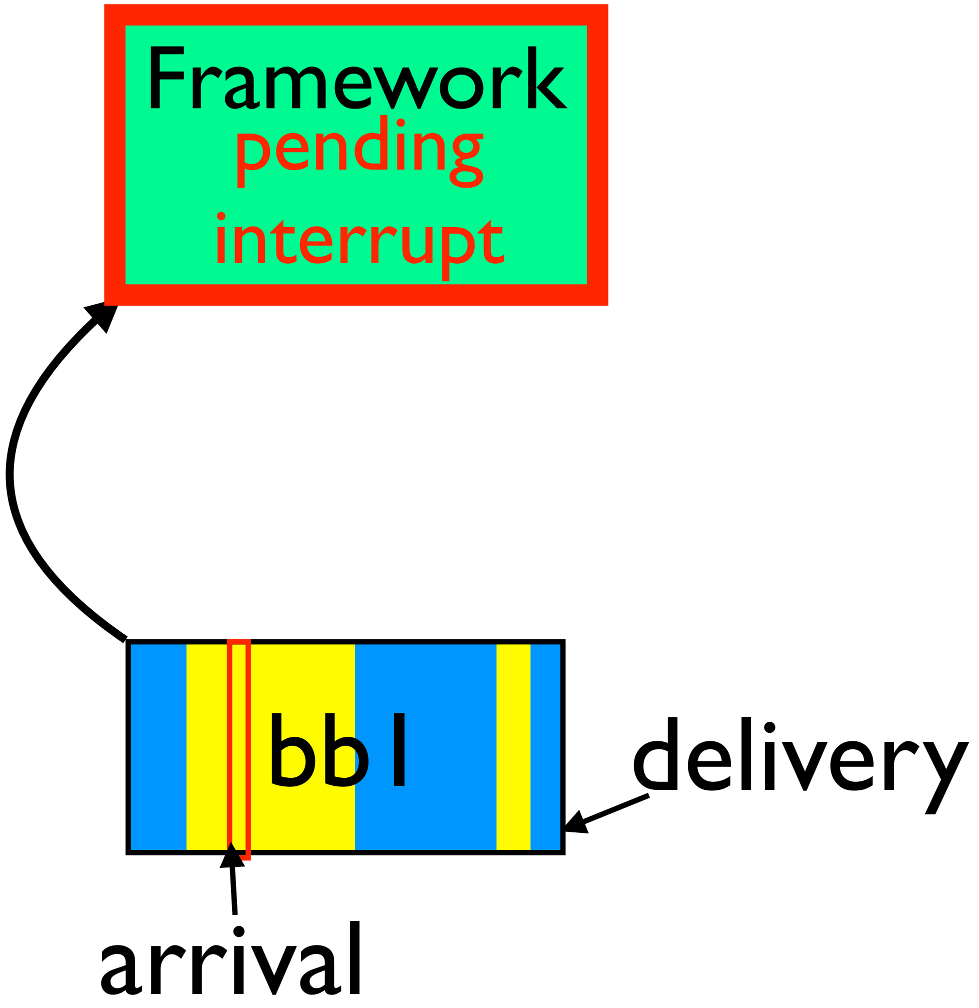


Delaying Interrupts

Where do we delay it until?

Delay until end of bb1?

- Avoids tail

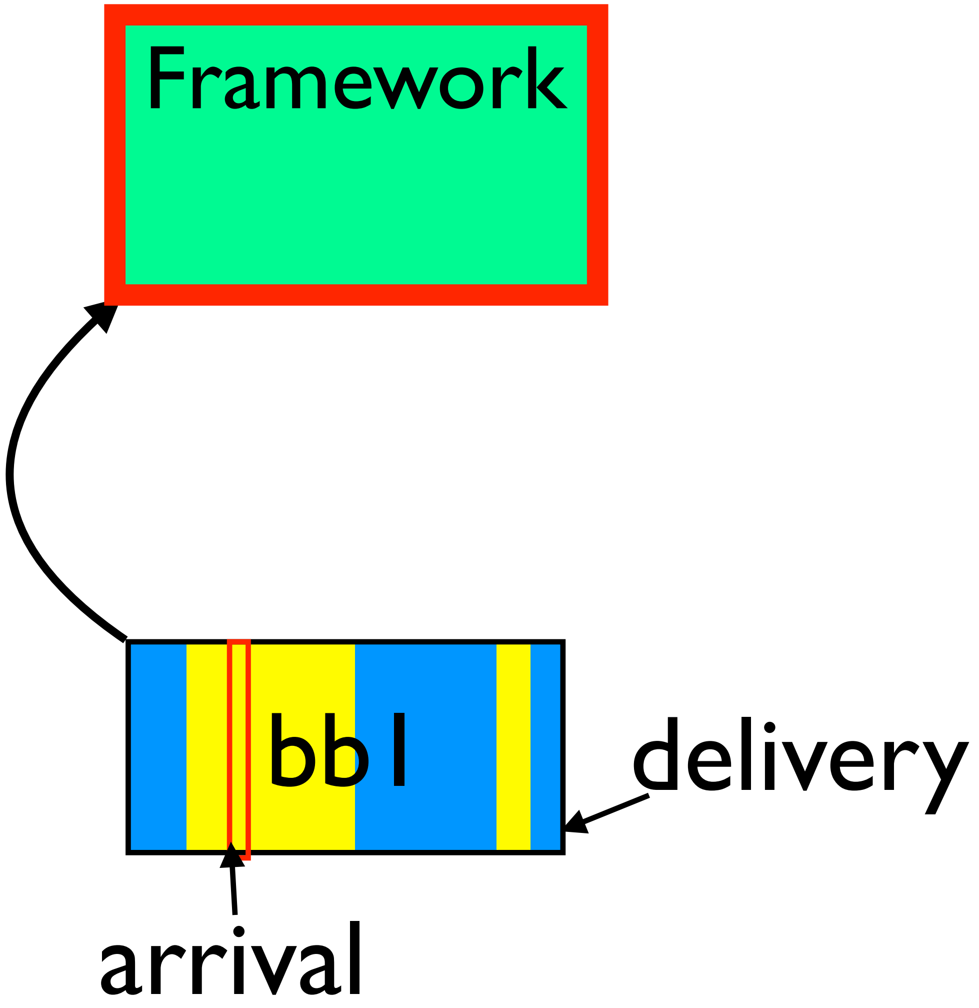


Delaying Interrupts

Where do we delay it until?

Delay until end of bb1?

- Avoids tail

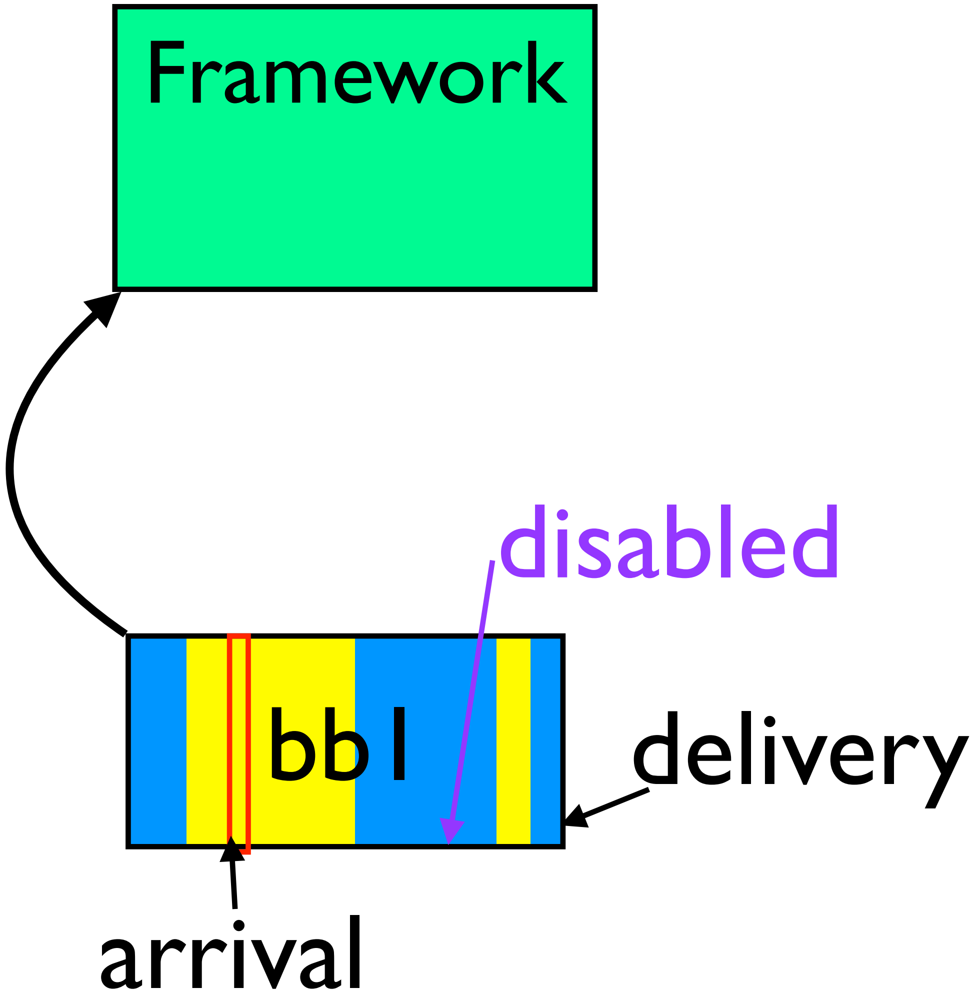


Delaying Interrupts

Where do we delay it until?

Delay until end of bb1?

- Avoids tail
- Problem if bb1 disables interrupt

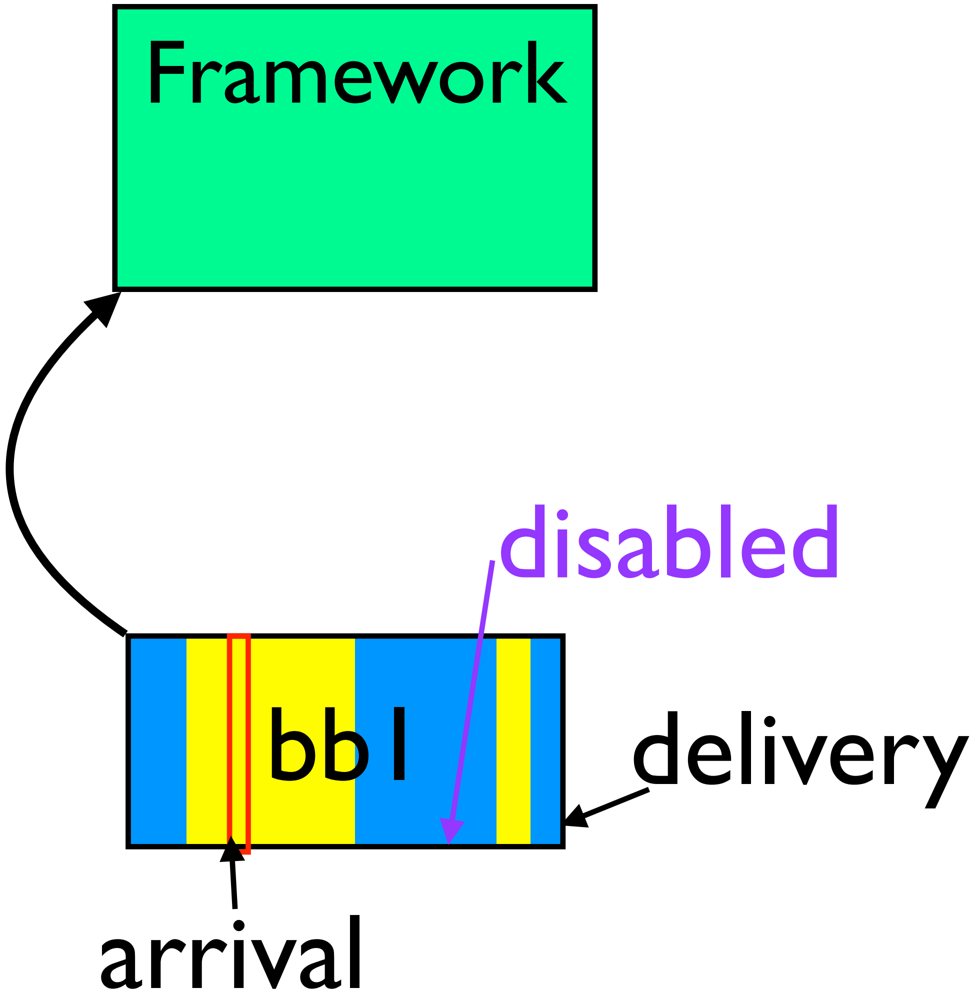


Delaying Interrupts

Where do we delay it until?

Delay until end of bb1?

- Avoids tail
- Problem if bb1 disables interrupt
  - Could be any MMIO
  - Framework cannot detect if enabled

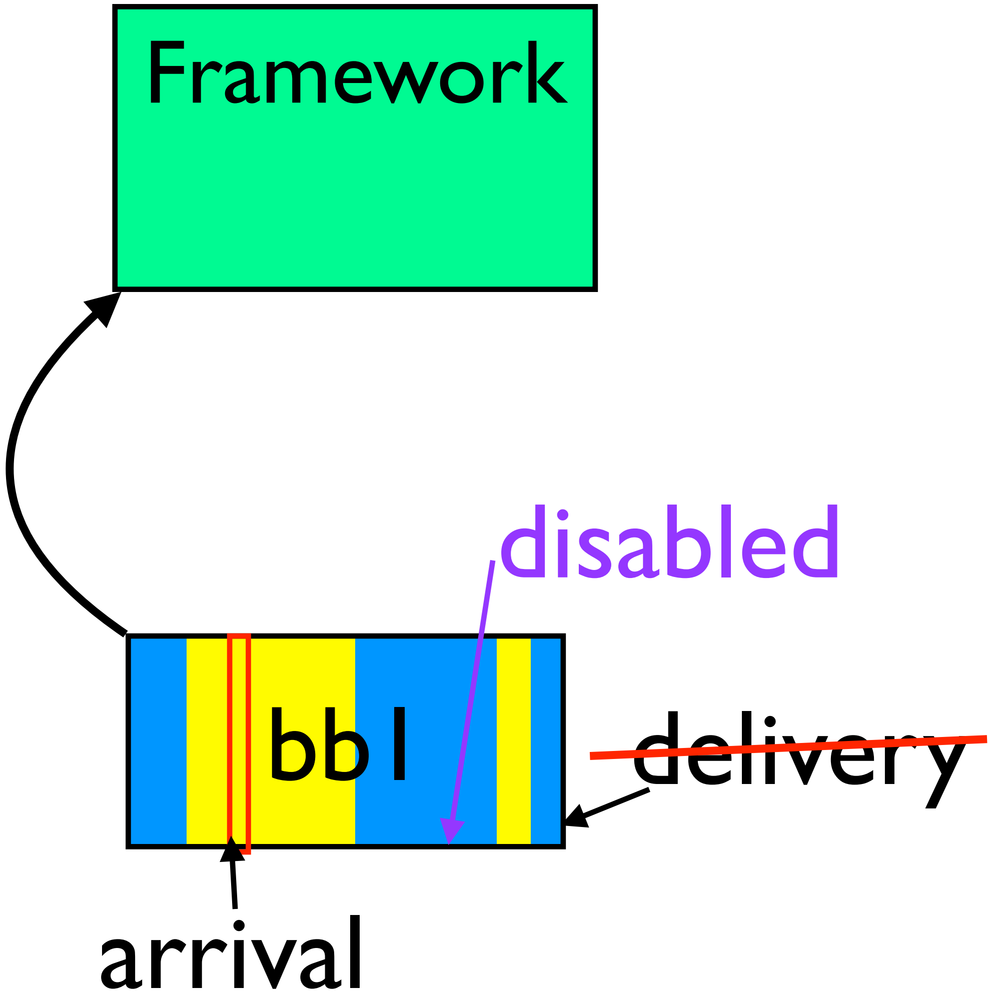


Delaying Interrupts

Where do we delay it until?

Delay until end of bb1? ✗

- Avoids tail
- Problem if bb1 disables interrupt
  - Could be any MMIO
  - Framework cannot detect if enabled



Framework

disabled

bb1

~~delivery~~

arrival

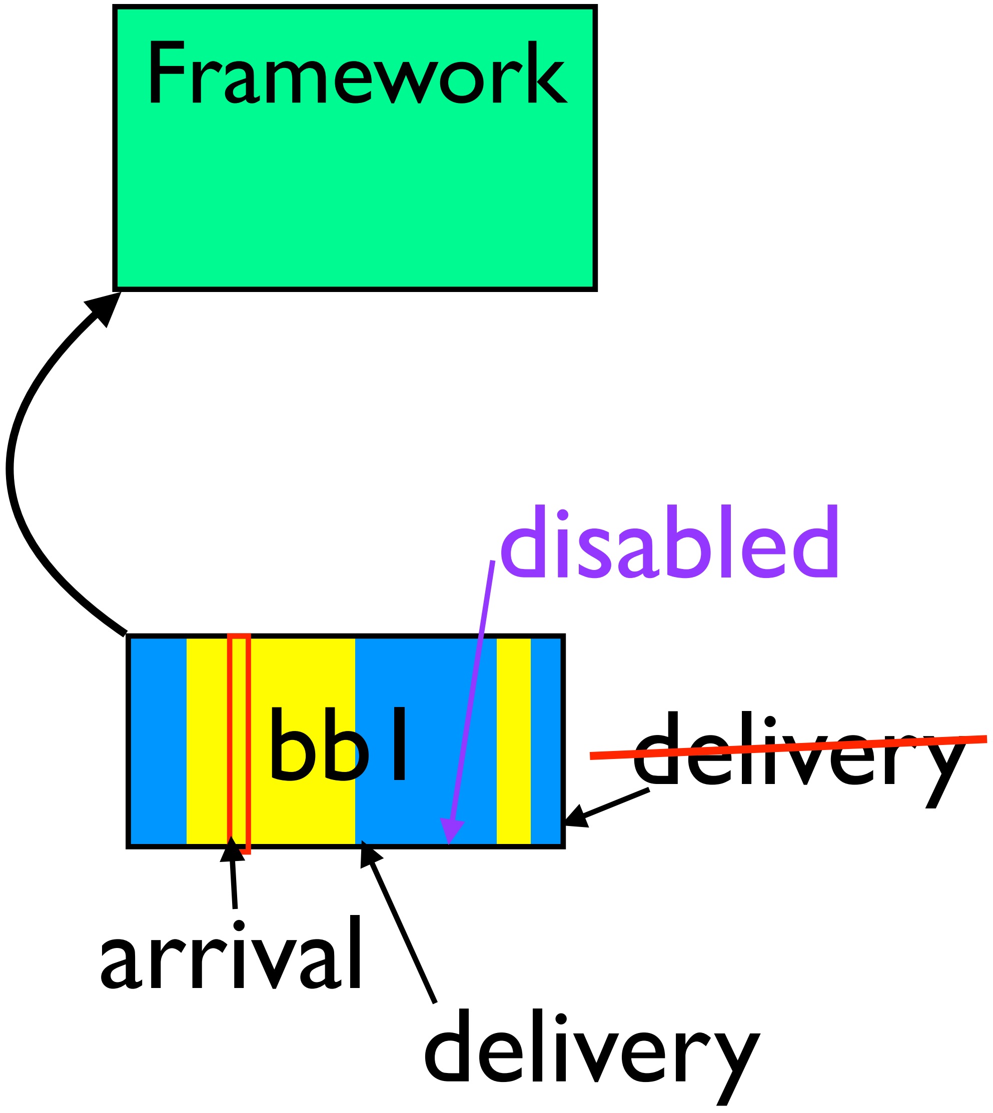
Delaying Interrupts

Where do we delay it until?

Delay until end of bb1? ✗

- Avoids tail
- Problem if bb1 disables interrupt
  - Could be any MMIO
  - Framework cannot detect if enabled

Must deliver before next OS instruction

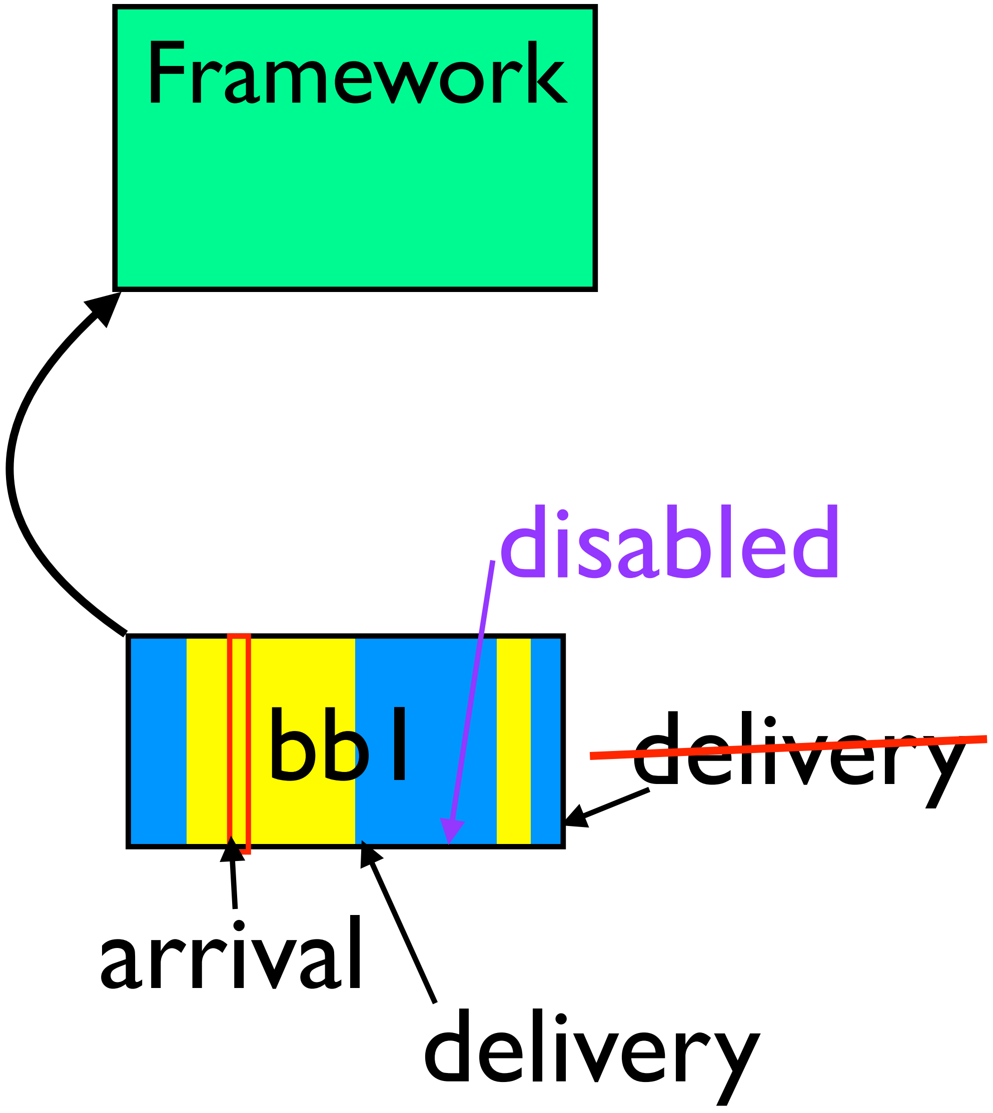


Delaying Interrupts

Where do we delay it until?

Delay until end of bb1? ✗

- Avoids tail
- Problem if bb1 disables interrupt
  - Could be any MMIO
  - Framework cannot detect if enabled



Must deliver before next OS instruction

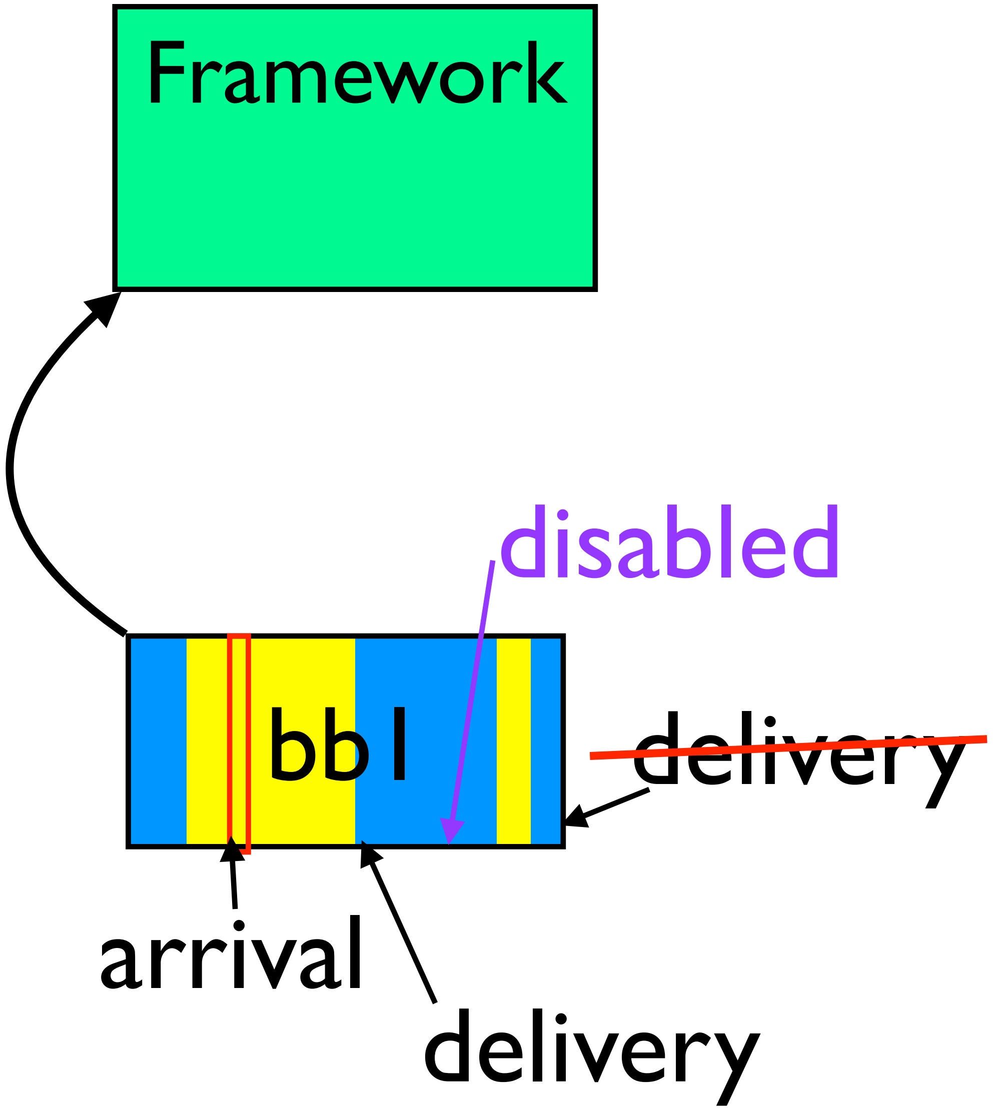
- Delay until end of instrumentation

Delaying Interrupts

Where do we delay it until?

Delay until end of bb1? ✗

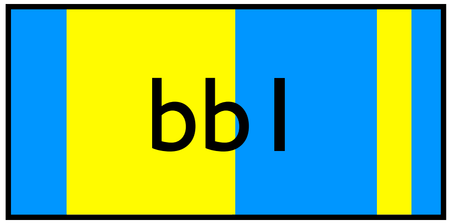
- Avoids tail
- Problem if bb1 disables interrupt
  - Could be any MMIO
  - Framework cannot detect if enabled



Must deliver before next OS instruction

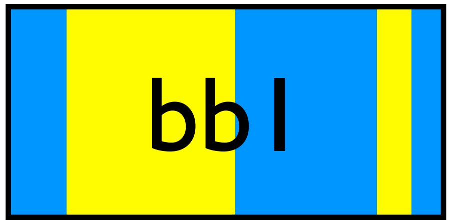
- Delay until end of instrumentation
- Still duplicates tail

How to Delay Interrupts



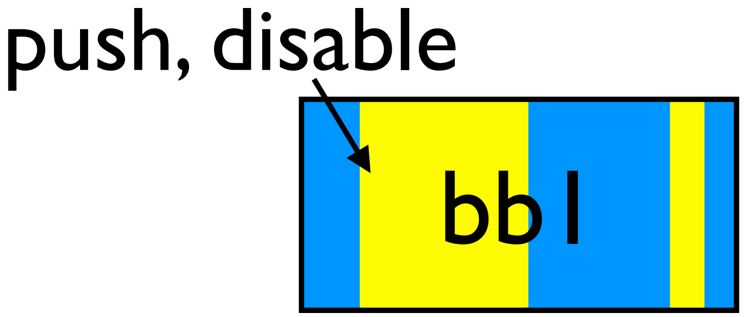
How to Delay Interrupts

Could disable them on the CPU



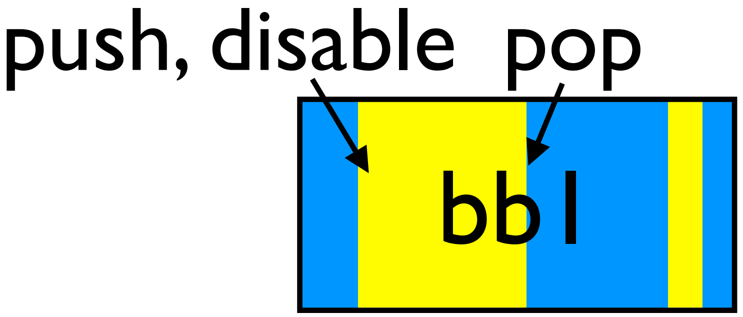
How to Delay Interrupts

Could disable them on the CPU



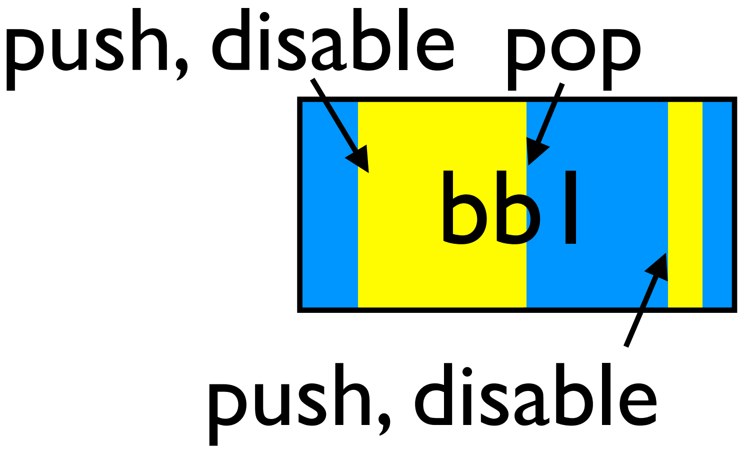
How to Delay Interrupts

Could disable them on the CPU



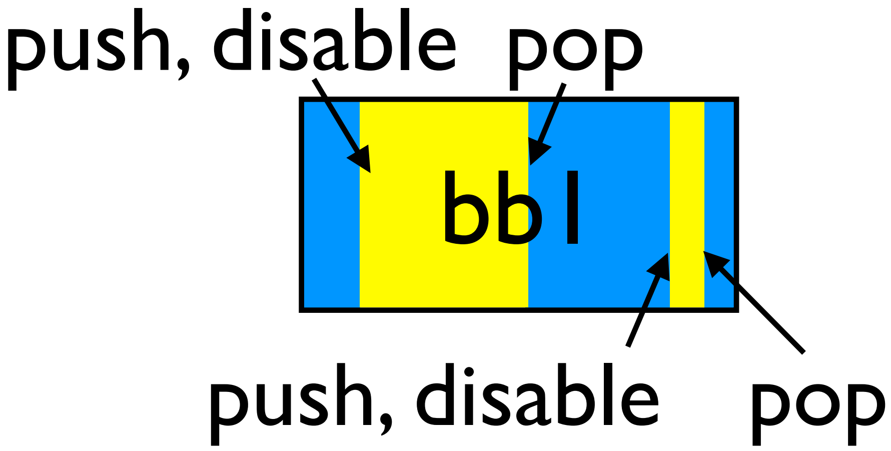
How to Delay Interrupts

Could disable them on the CPU



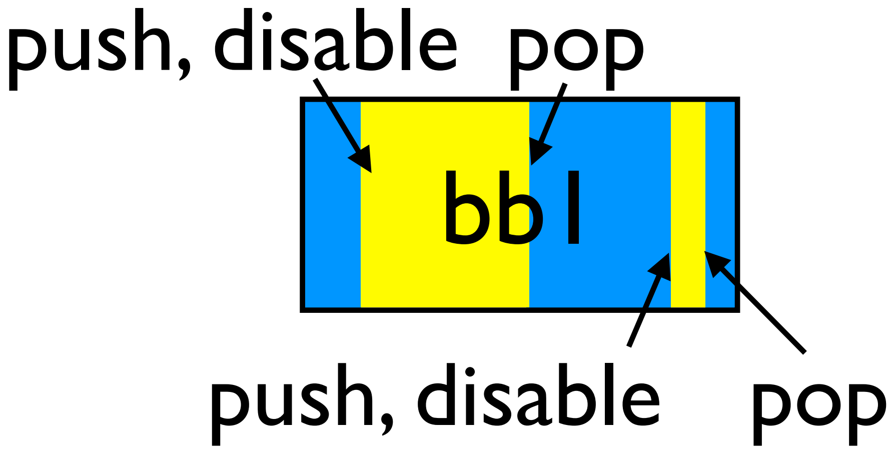
How to Delay Interrupts

Could disable them on the CPU



How to Delay Interrupts

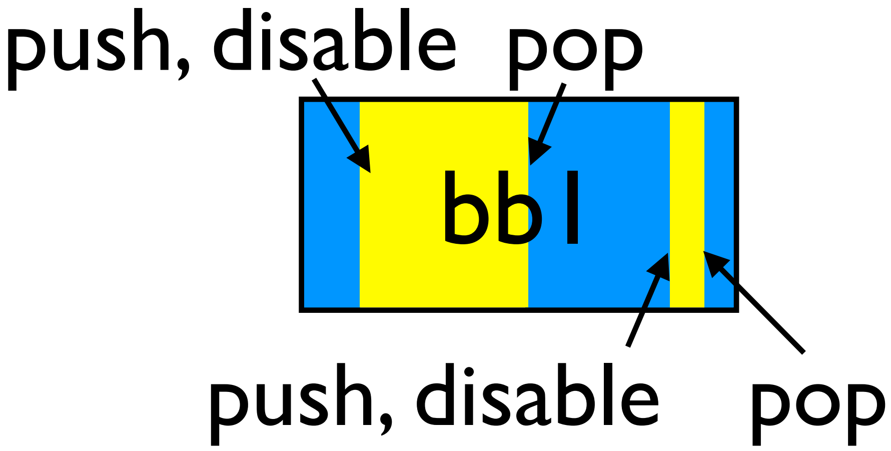
Could disable them on the CPU



But performance would be bad

How to Delay Interrupts

Could disable them on the CPU



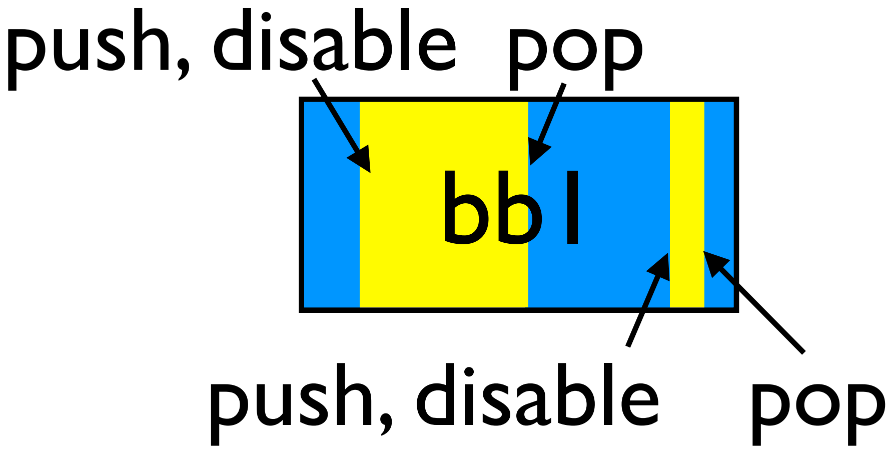
But performance would be bad

Instead, have framework handle it

- Extra overhead for interrupt, cheaper instrumentation

How to Delay Interrupts

Could disable them on the CPU



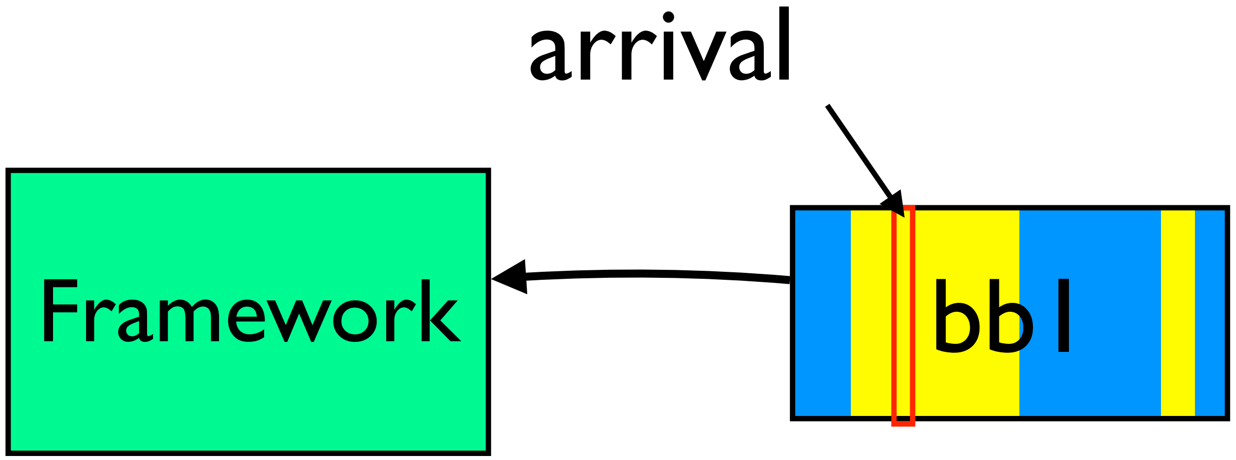
But performance would be bad

Instead, have framework handle it

- Extra overhead for interrupt, cheaper instrumentation
- Instrumentation more frequent than interrupts
- Gigabit NIC sends interrupt every 100μs ≈ 100K instr.

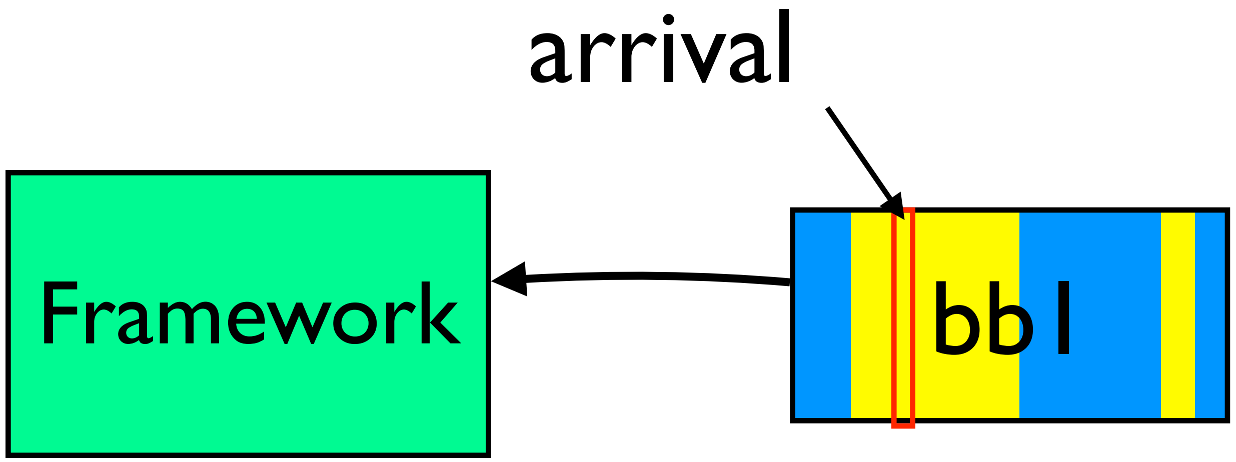
Delaying with Patches

Example: interrupt 239



Delaying with Patches

Example: interrupt 239

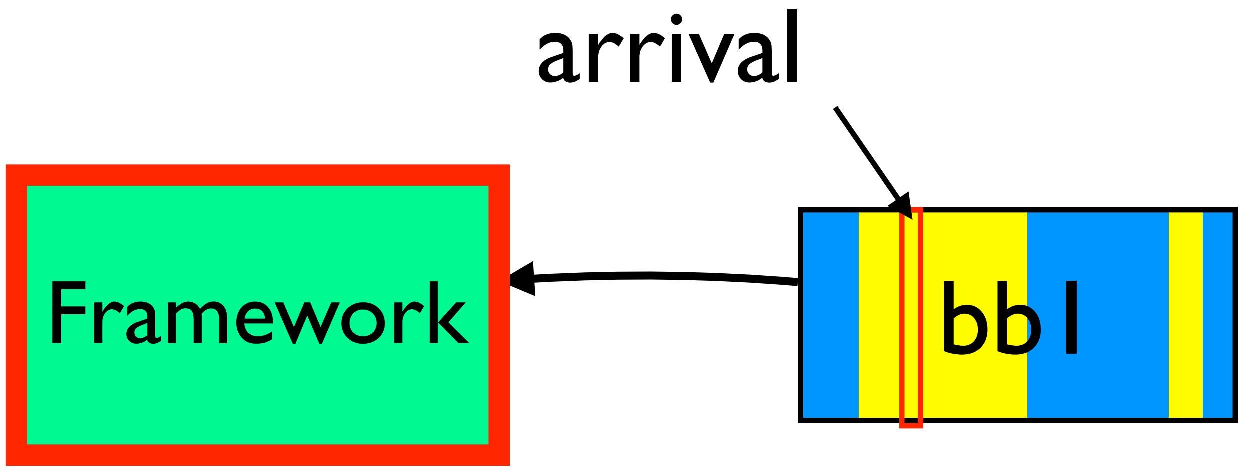


Interrupt Stack Frame

Interrupts enabled	yes
...	

Delaying with Patches

Example: interrupt 239



Interrupt Stack Frame

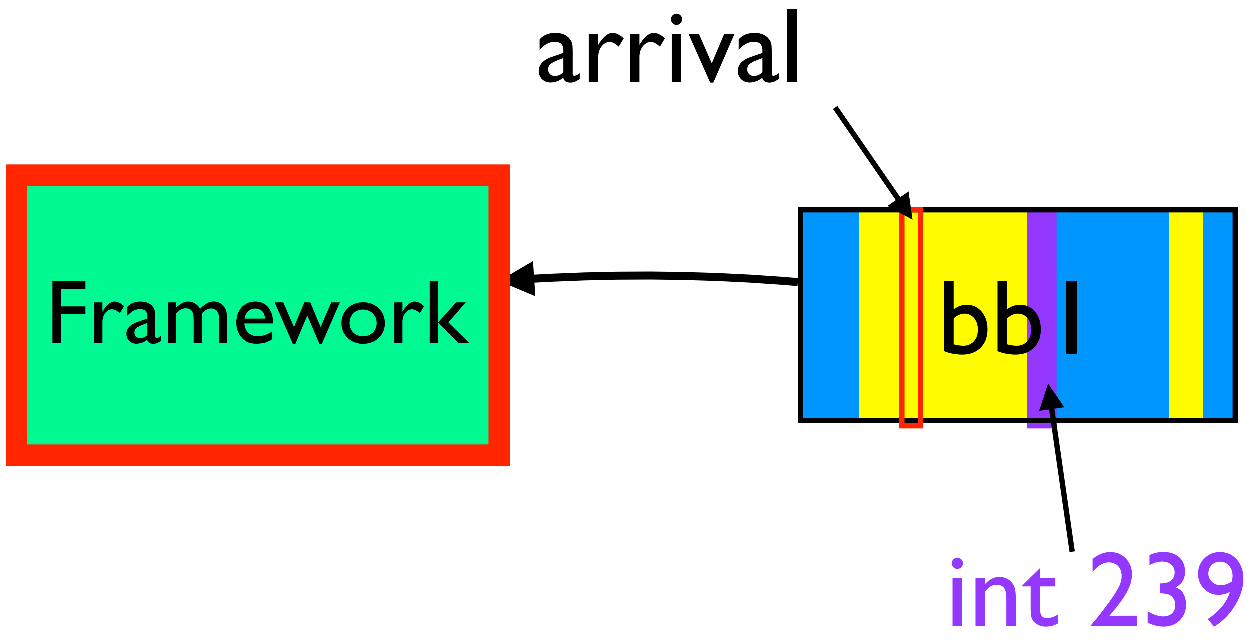
Interrupts enabled	yes
...	

Delaying with Patches

Example: interrupt 239

The framework

1. Patches next native instruction

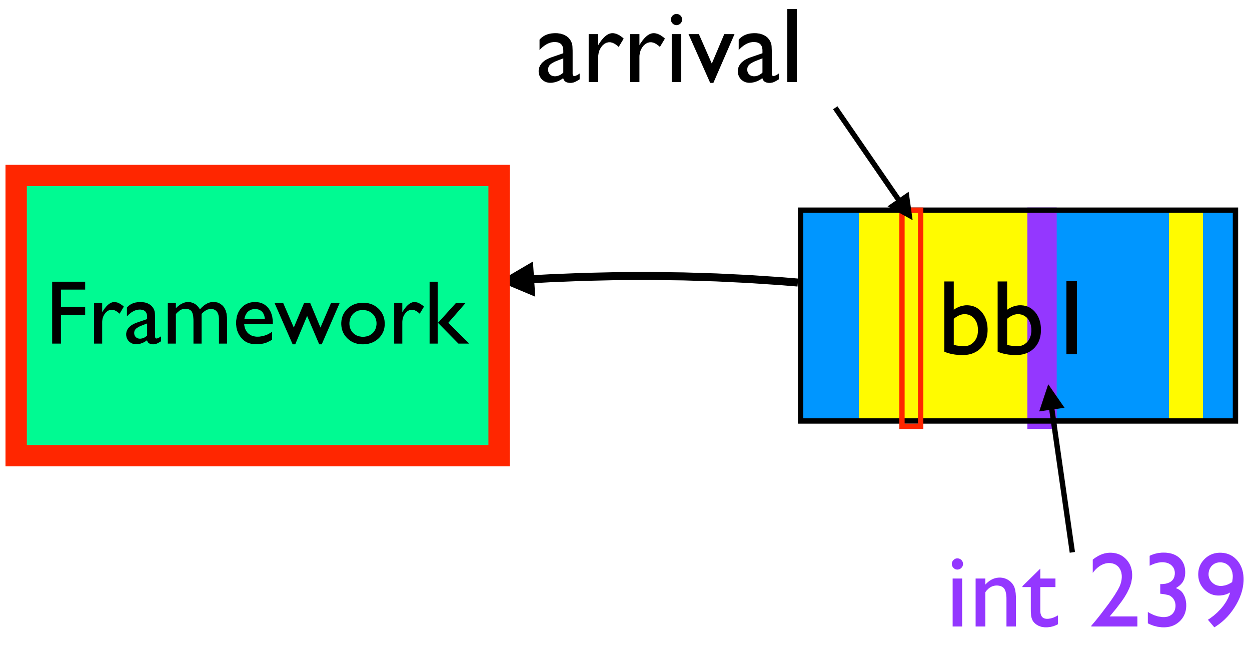


Interrupt Stack Frame

Interrupts enabled	yes
...	

Delaying with Patches

Example: interrupt 239



The framework

1. Patches next native instruction
2. Disables interrupts on iret

Interrupt Stack Frame

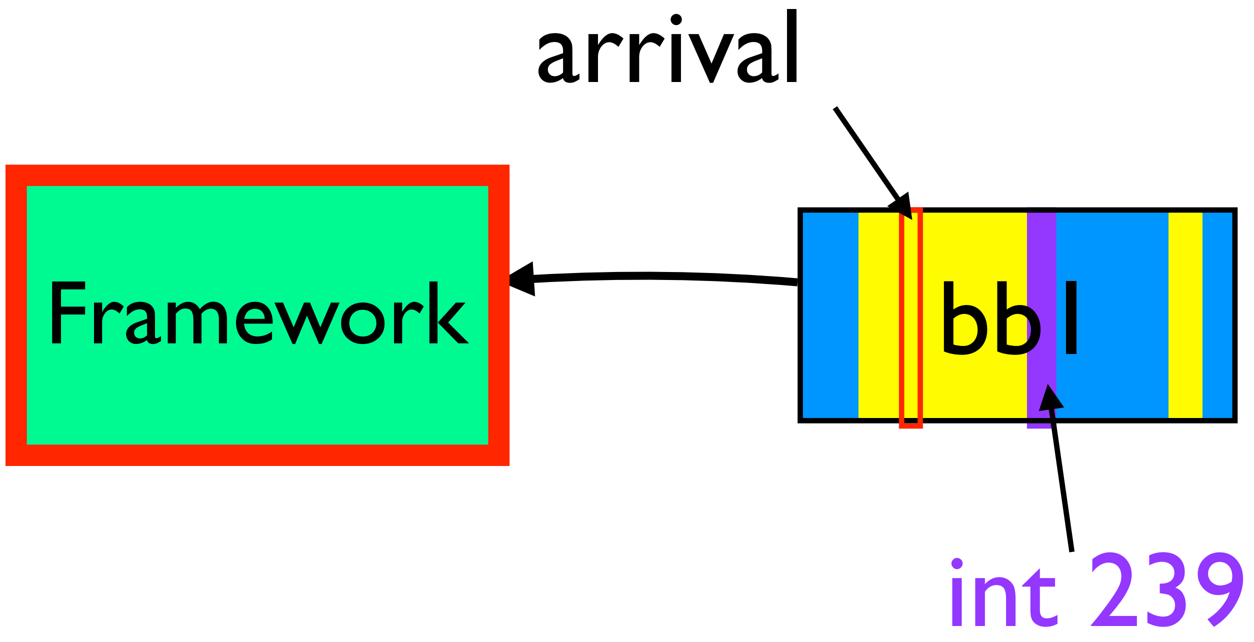
Interrupts enabled	~~yes~~ no
...	

Delaying with Patches

Example: interrupt 239

The framework

1. Patches next native instruction
2. Disables interrupts on iret
3. iret



Interrupt Stack Frame

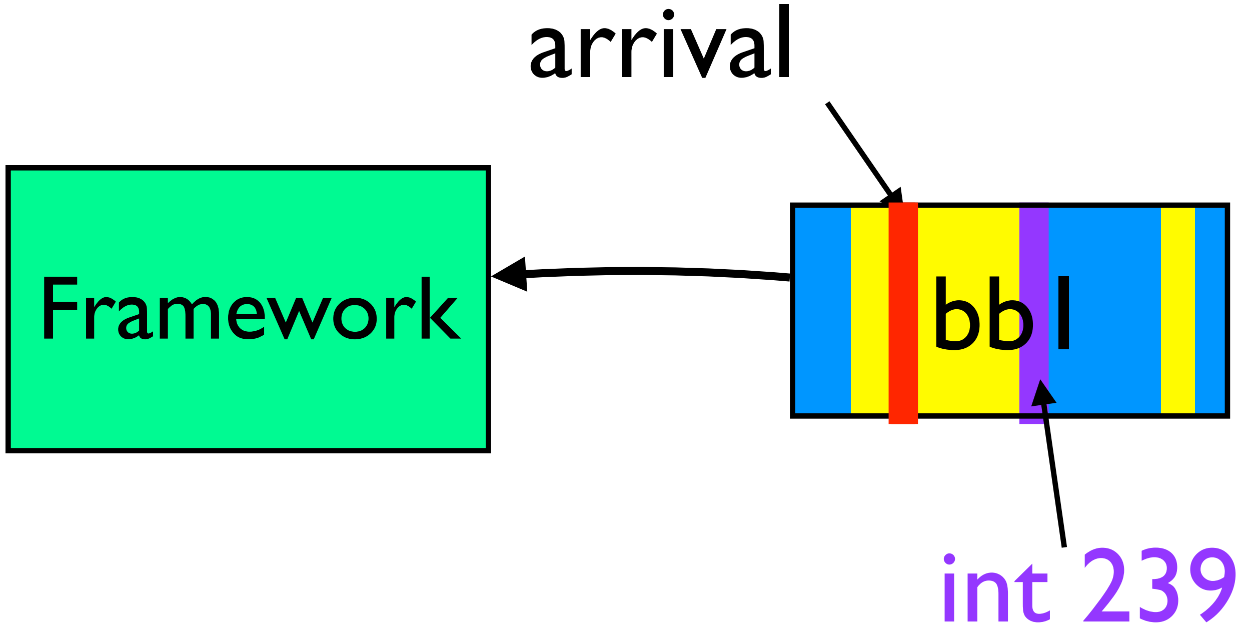
Interrupts enabled	~~yes~~ no
...	

Delaying with Patches

Example: interrupt 239

The framework

1. Patches next native instruction
2. Disables interrupts on iret
3. iret



Interrupt Stack Frame

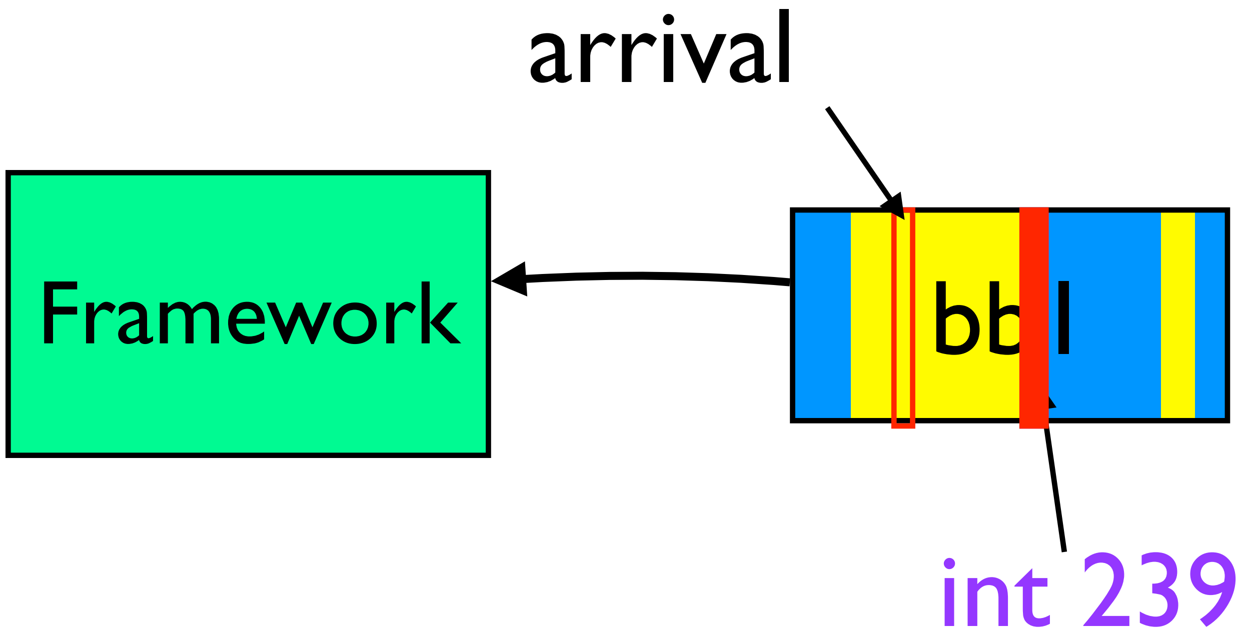
Interrupts enabled	~~yes~~ no
...	

Delaying with Patches

Example: interrupt 239

The framework

1. Patches next native instruction
2. Disables interrupts on iret
3. iret

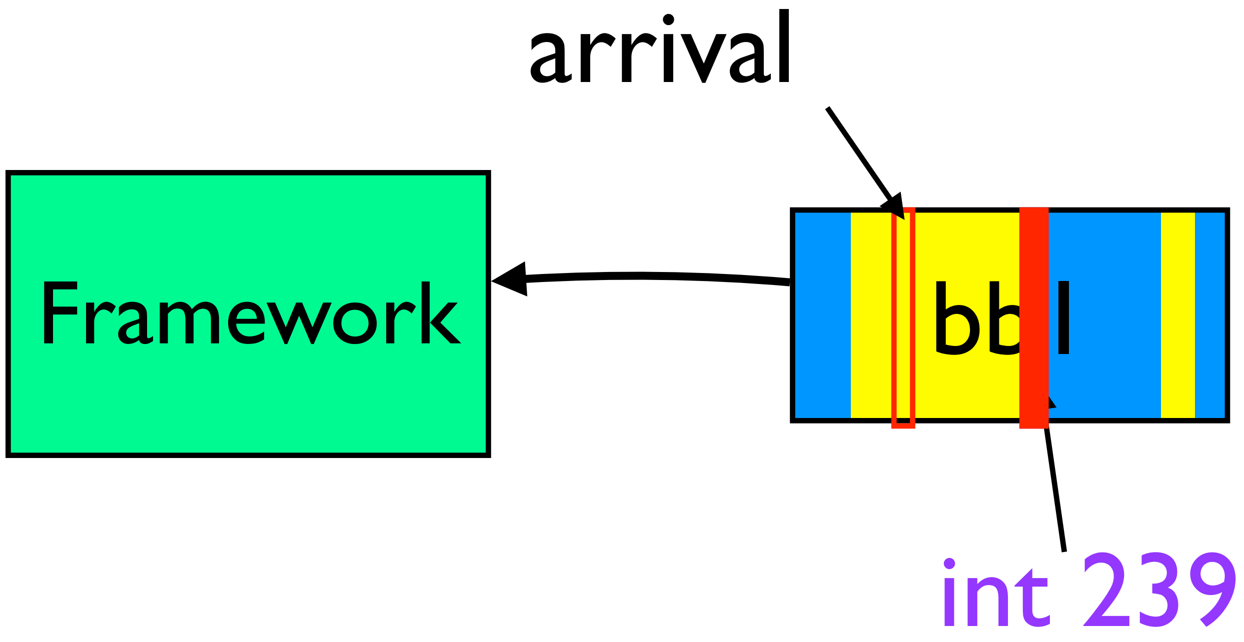


Interrupt Stack Frame

Interrupts enabled	~~yes~~ no
...	

Delaying with Patches

Example: interrupt 239



The framework

1. Patches next native instruction
2. Disables interrupts on iret
3. iret

Interrupt Stack Frame

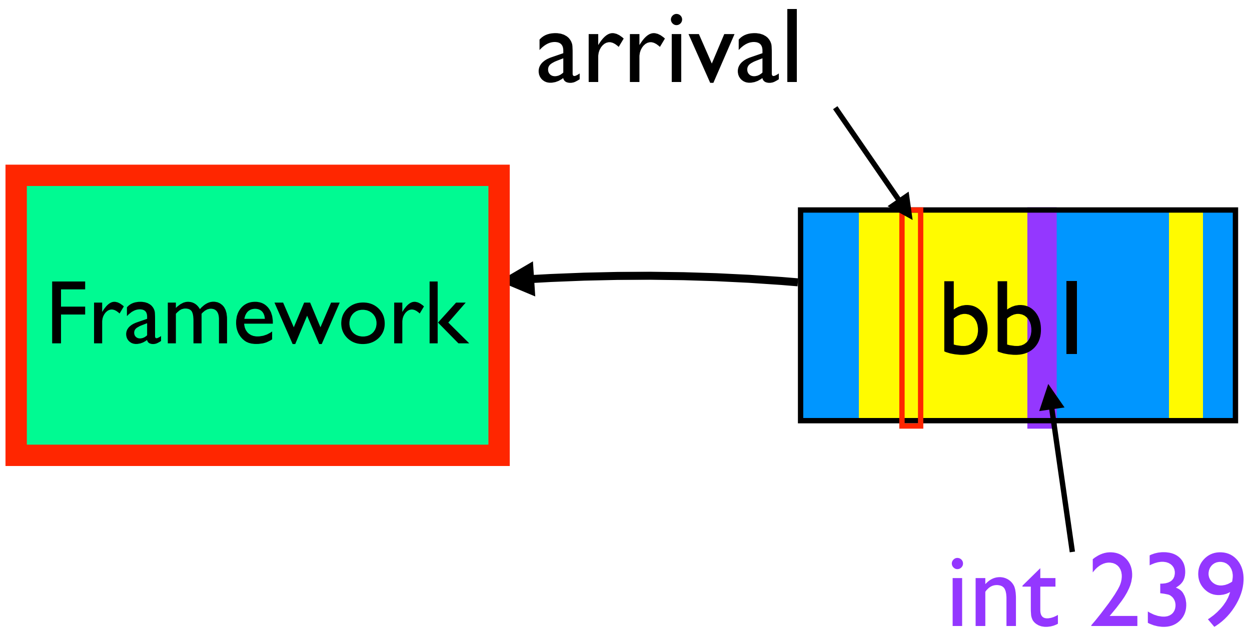
Interrupts enabled	no ~~yes~~
...	

Patch Interrupt Stack Frame

Interrupts enabled	no
...	

Delaying with Patches

Example: interrupt 239



The framework

1. Patches next native instruction
2. Disables interrupts on iret
3. iret

Interrupt Stack Frame

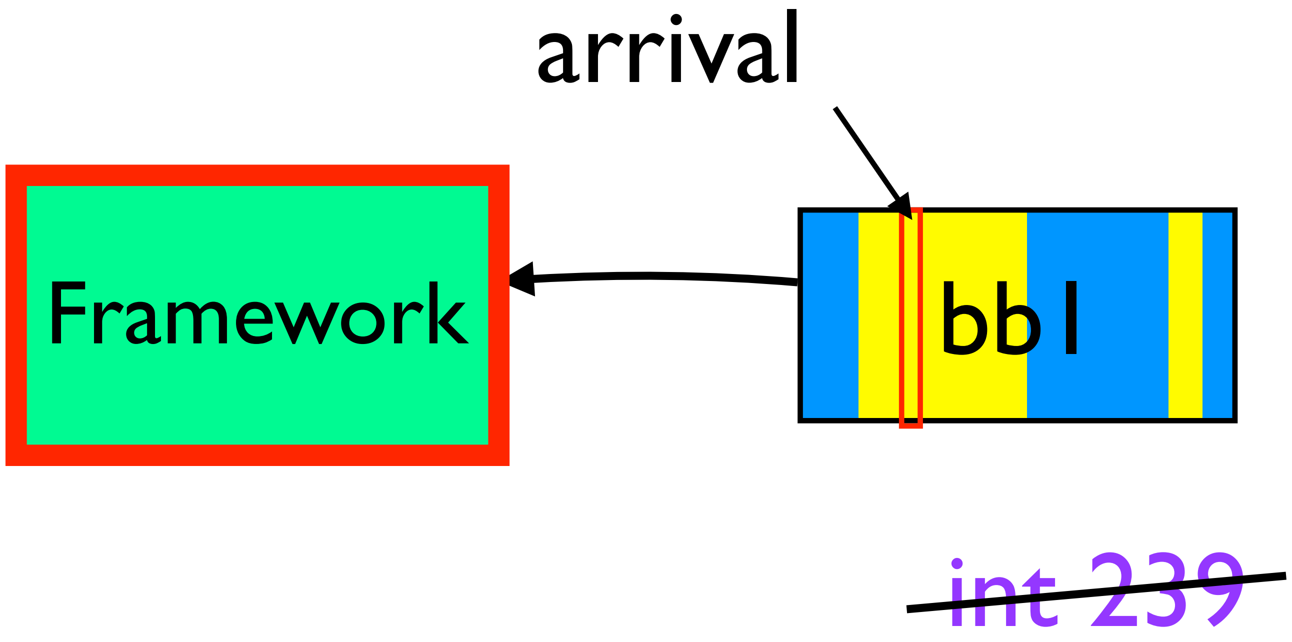
Interrupts enabled	~~yes~~ no
...	

Patch Interrupt Stack Frame

Interrupts enabled	no
...	

Delaying with Patches

Example: interrupt 239



The framework

1. Patches next native instruction
2. Disables interrupts on iret
3. iret
4. Removes patch

Interrupt Stack Frame

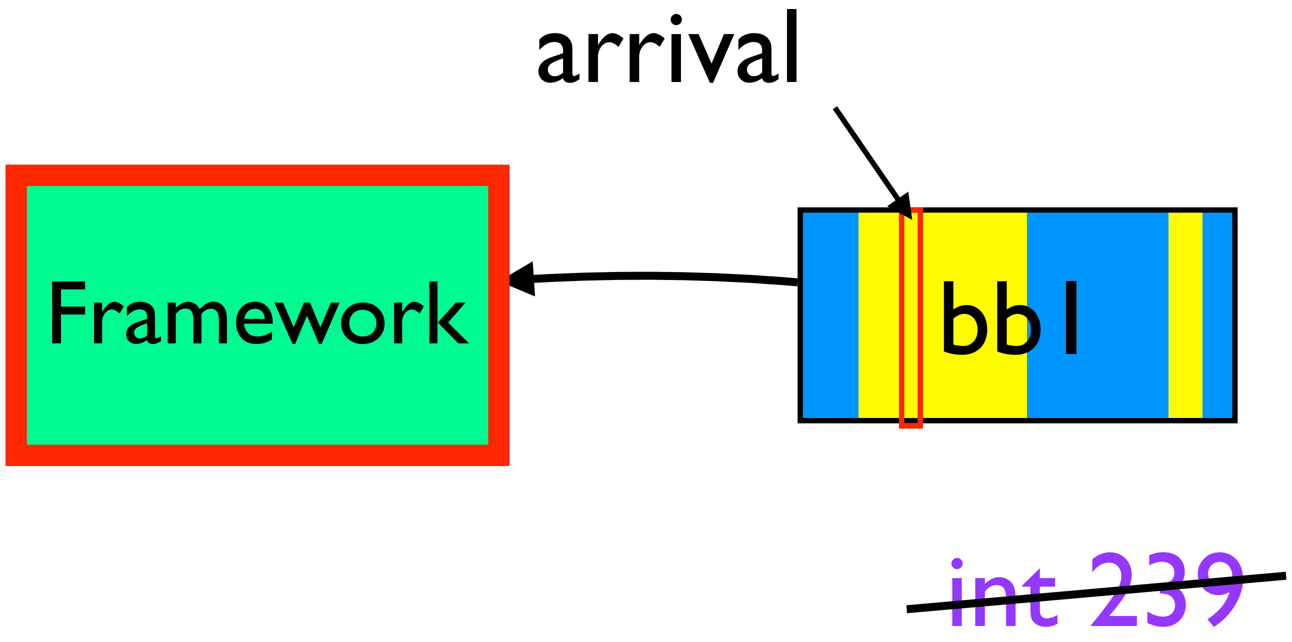
Interrupts enabled	no ~~yes~~
...	

Patch Interrupt Stack Frame

Interrupts enabled	no
...	

Delaying with Patches

Example: interrupt 239



The framework

1. Patches next native instruction
2. Disables interrupts on iret
3. iret
4. Removes patch
5. Enables interrupts on iret

Interrupt Stack Frame

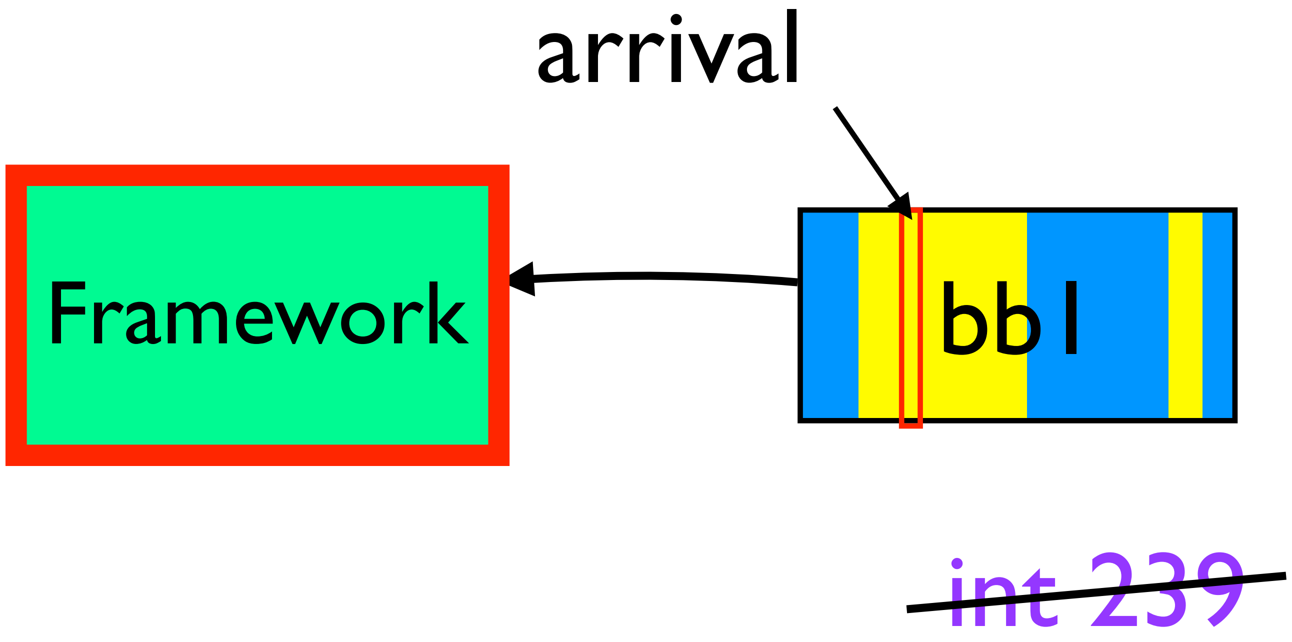
Interrupts enabled	~~yes~~ no
…	

Patch Interrupt Stack Frame

Interrupts enabled	~~no~~ yes
…	

Delaying with Patches

Example: interrupt 239



The framework

1. Patches next native instruction
2. Disables interrupts on iret
3. iret
4. Removes patch
5. Enables interrupts on iret
6. Run instrumented interrupt handler

Interrupt Stack Frame

Interrupts enabled	~~yes~~ no
...	

Patch Interrupt Stack Frame

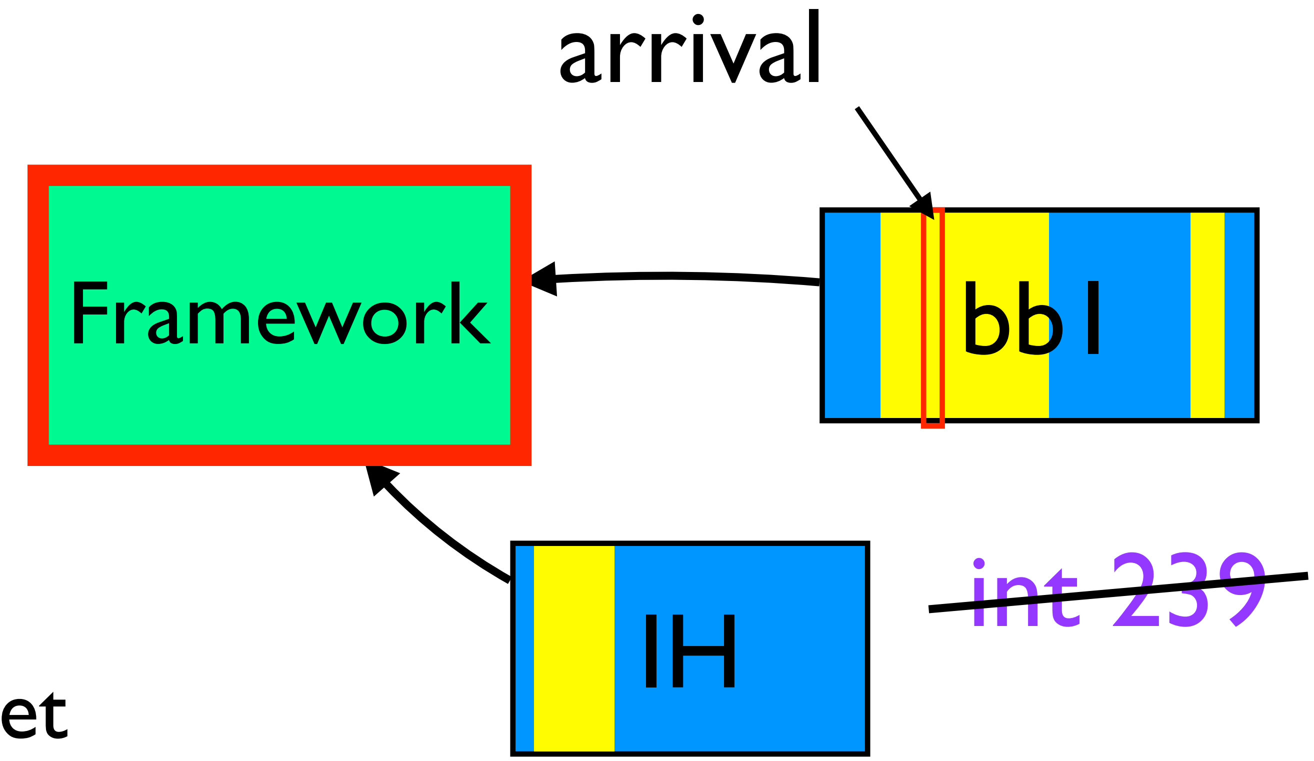
Interrupts enabled	~~no~~ yes
...	

Delaying with Patches

Example: interrupt 239

The framework

1. Patches next native instruction
2. Disables interrupts on iret
3. iret
4. Removes patch
5. Enables interrupts on iret
6. Run instrumented interrupt handler



arrival

Framework

bb1

IH

~~int 239~~

Interrupt Stack Frame

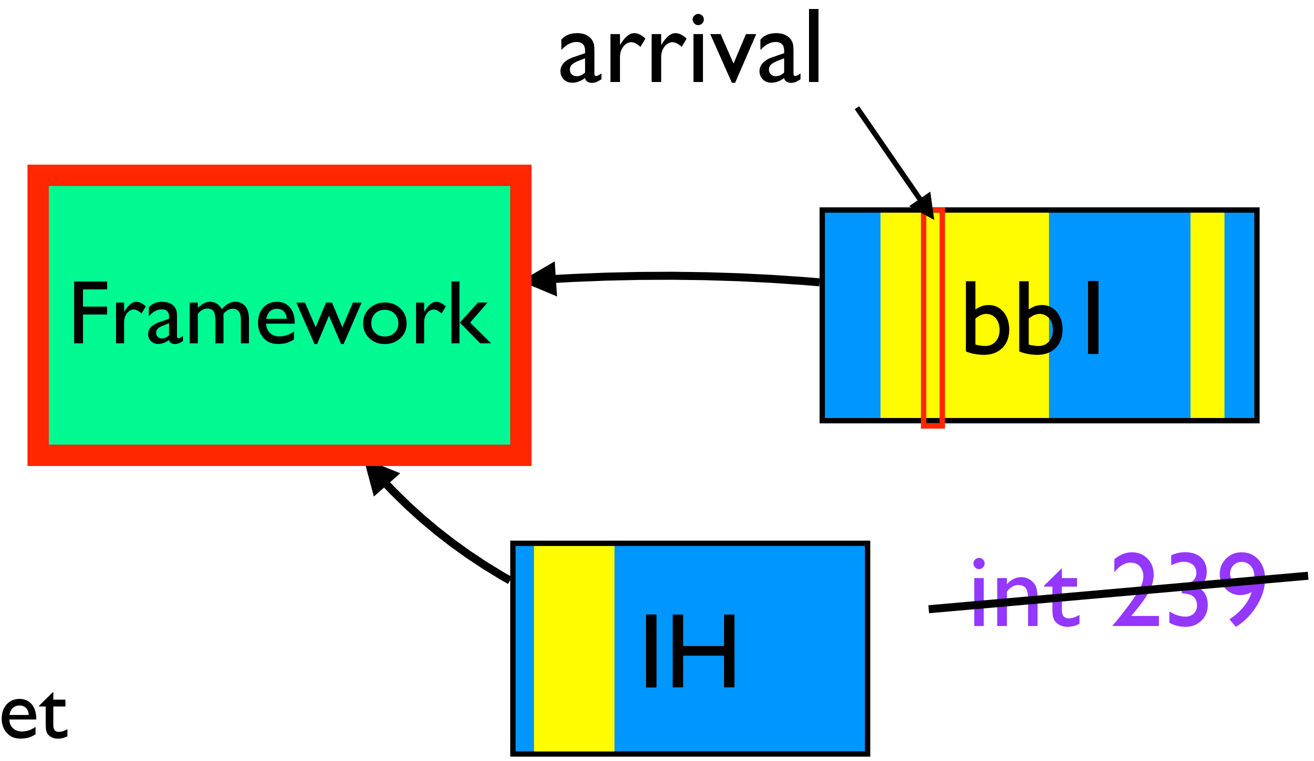
Interrupts enabled	~~yes~~ no
...	

Patch Interrupt Stack Frame

Interrupts enabled	~~no~~ yes
...	

Delaying with Patches

Example: interrupt 239



The framework

1. Patches next native instruction
2. Disables interrupts on iret
3. iret
4. Removes patch
5. Enables interrupts on iret
6. Run instrumented interrupt handler

Interrupt Stack Frame

Interrupts enabled	~~yes~~ no
...	

Patch Interrupt Stack Frame

Interrupts enabled	~~no~~ yes
...	

Okay, what's the performance?

Performance

Ran framework with instruction counting tool

- Intel Quad Core i7 2.8Ghz, 8GB, 64-bit Ubuntu 10.10

Low application overhead

- JavaScript, Mozilla Kraken: 3% overhead
- Parallel Linux kernel compile: 30% overhead
  - 18% user time increase
  - 143% system time increase

Overhead commensurate with OS activity

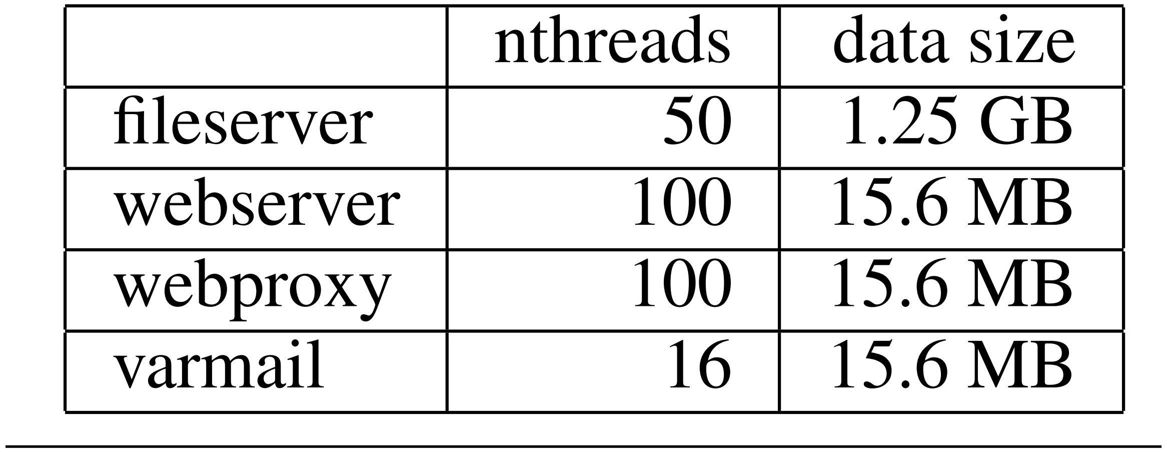
- How bad can this get?

Stress Test Setup

Apachebench and Filebench

Configured benchmarks to stress CPUs and kernel

- Large buffer cache - no disk I/O
- Many threads - lots of context switching
- 100% utilization - shows interrupt processing overhead



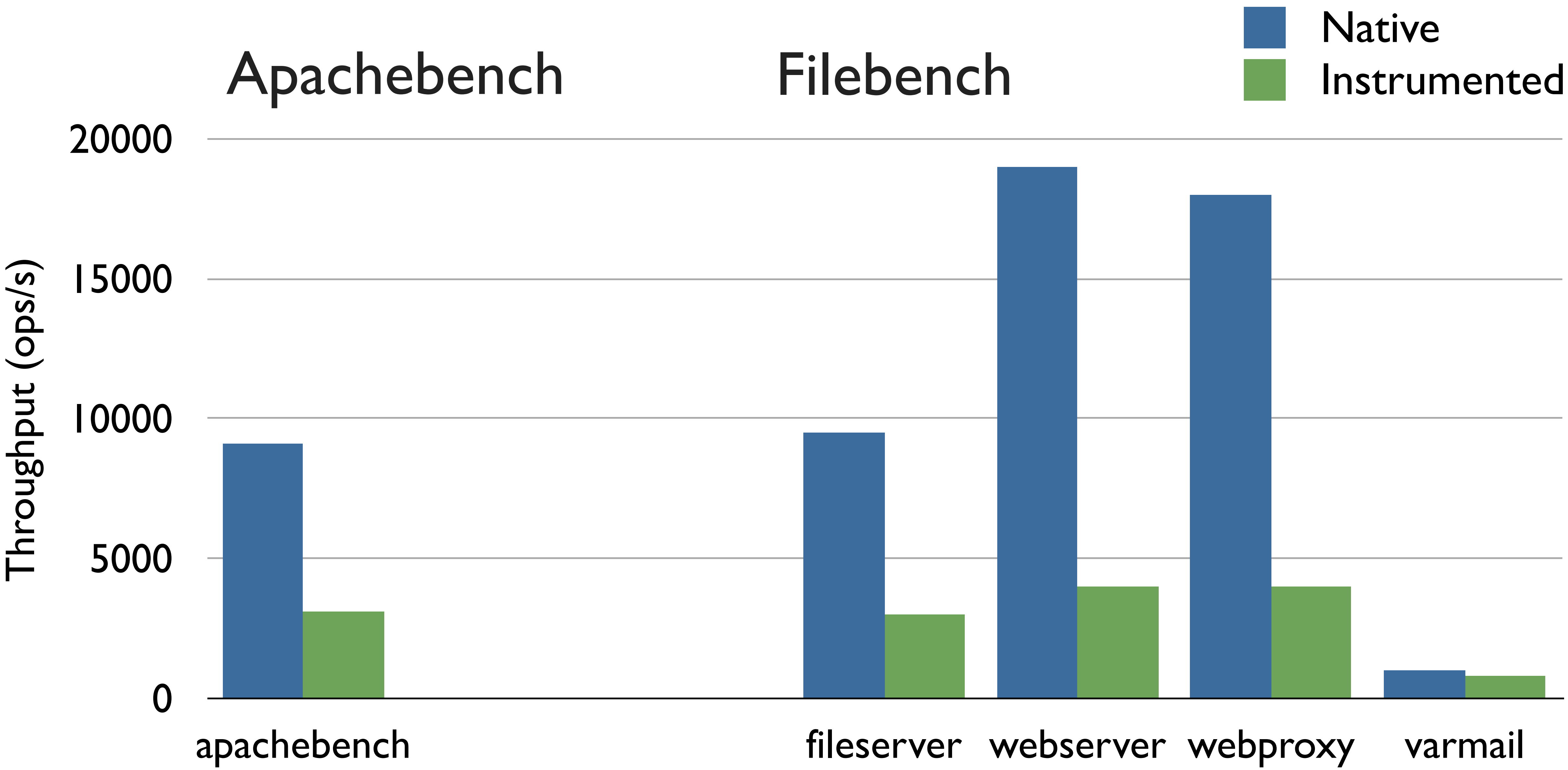
	nthreads	data size
fileserver	50	1.25 GB
webserver	100	15.6 MB
webproxy	100	15.6 MB
varmail	16	15.6 MB

Table 1. Filebench parameters

concurrency level 200

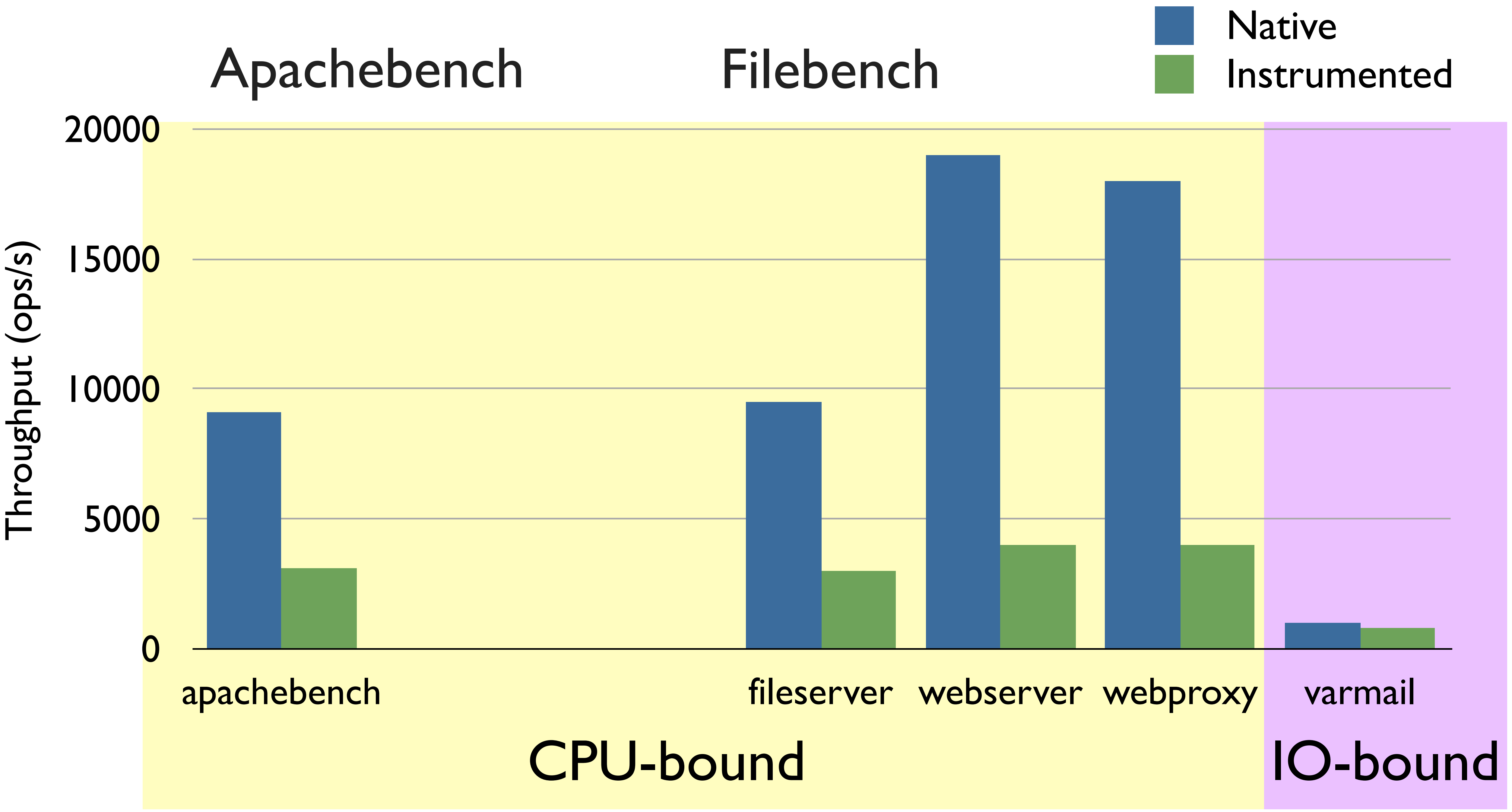
Apachebench Parameters

Stress Test Results



Less than 5x - Reasonable overhead for debugging tools

Stress Test Results



Less than 5x - Reasonable overhead for debugging tools

Summary

Enables dynamic binary instrumentation of OS

Makes it easy to write complex instrumentation

Built useful memory checking tools

Works with arbitrary devices & drivers

Questions?